

User's Manual



A Revolution in OCR Software

Microsoft, Windows, Windows NT, Windows 2000, Windows XP, Visual Basic, Microsoft .NET, Visual C++, Visual C#, and ActiveX are either trademarks or registered trademarks of Microsoft Corporation.

The Sentry Spelling-Checker Engine Copyright © 2002 Wintertree Software Inc.

All other nationally and internationally recognized trademarks and tradenames are hereby recognized.

Copyright © 2005-2008 by MVTec Software GmbH, München, Germany



Edition 1	January 2005	(ActivVisionTools 3.0)
Edition 2	February 2006	(ActivVisionTools 3.1)
Edition 3	May 2008	(ActivVisionTools 3.2)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

More information about ActivVisionTools can be found at:

<http://www.activ-vision-tools.com>

How to Read This Manual

This manual explains how to use ActivOCR to read text from images. It describes the functionality of ActivOCR and its cooperation with other ActivVisionTools with Visual Basic examples. Before reading this manual, we recommend to read the manual [Getting Started with ActivVisionTools](#), which introduces the basic concepts of ActivVisionTools, and the [User's Manual for ActivView](#) to learn how to load and display images.

For each example in this manual, there is a corresponding Visual Basic project; these projects can be found in the subdirectory `examples\manuals\activocr` of the ActivVisionTools base directory you chose during installation (default: `C:\Program Files\MVTec\ActivVisionTools`). Of course, you can also create your own Visual Basic projects from scratch. Note, that the screenshots in the manual may differ slightly from the corresponding Visual Basic projects. To follow the examples actively, first install and configure ActivVisionTools as described in the manual [Getting Started with ActivVisionTools](#).

We recommend to **create a private copy of the example projects** because by experimenting with the projects, you also change their *state*, which is then automatically stored in the so-called description files (extension `.dsc`) by ActivVisionTools. Of course, you can restore the state of a project by retrieving the corresponding description file from the CD.



Contents

1	About ActivOCR	1
1.1	Introducing ActivOCR	2
1.2	The Sub-Tools of ActivOCR	4
1.3	The Sub-Tools of ActivWordProcess	8
2	Using ActivOCR	11
2.1	Specifying the Search Area	12
2.2	Extracting the Symbols from the Image	14
2.3	Selecting and Creating OCR Fonts	22
2.4	Displaying and Selecting Results	38
3	Combining ActivOCR with other ActivVisionTools	41
3.1	Improving and Checking OCR Results Using ActivWordProcess	42
3.2	Evaluating Results	52
3.3	Output of Results	54
4	Tips & Tricks	57
4.1	Adapting the Display of Results	58
4.2	Configuring the Two Execution Modes	60
4.3	Accessing Results Via the Programming Interface	62
4.4	Extracting Symbols Using HALCON	70
4.5	Tuning the OCR Classifier	74

4.6	Questions & Answers	76
A	Segmentation Methods	79
A.1	Segmentation Methods	80
A.2	Partitioning	82
A.3	The Effect of the Parameter Settings	85
B	Ready-to-Use OCR Fonts	93
B.1	Nomenclature	94
B.2	Document	94
B.3	DotPrint	95
B.4	HandWritten	95
B.5	Industrial	96
B.6	MICR	96
B.7	OCRA	97
B.8	OCRB	98
B.9	Pharma	99
C	Advanced OCR Font Setting Features	101
C.1	Description of the Features	102
D	The Position Code	109
D.1	Definition	110
D.2	Examples	111
E	Syntax of ActivWordProcess	113
E.1	File Format	114
E.2	Examples for Syntax Strings	115
F	Files and Directories	117

F.1	Files	118
F.2	Directories	118

Chapter 1

About ActivOCR

This chapter will introduce you to the features and the basic concepts of ActivOCR and ActivWordProcess. It gives an overview about their *master tools* and the respective *support tools*, which are described in more detail in [chapter 2](#) on page 11 and [chapter 3](#) on page 41.

1.1	Introducing ActivOCR	2
1.2	The Sub-Tools of ActivOCR	4
1.3	The Sub-Tools of ActivWordProcess	8

1.1 Introducing ActivOCR

With ActivOCR, you can read text from images. A typical application can be set up simply by specifying a search area (ROI), adjusting some parameters for the extraction of the symbols, and selecting an appropriate OCR font. If the syntax of the words that must be read is known, or if they must match the entries of a dictionary, the results of ActivOCR can be further improved using `ActivWordProcess`.

In this manual, a *symbol* denotes an extracted region from which one or more *characters* are to be read. Typically, a symbol corresponds to a single character, but it can also correspond to more than one character, e.g., to represent ligatures (fi, ff, ...).

Automatic Reading

As soon as an ROI is defined, ActivOCR tries to read the contents of the ROI. For this, the symbols are extracted from the image and read according to the selected OCR font. The results are displayed in the image, directly below the extracted symbols.

The Search Area (ROI)

Specify at least one search area (ROI) for AVTOCR with `AVTViewROI`. The ROI must contain the text that is to be read.

If you want to read text which moves from image to image, the ROI must be moved accordingly. For this, you need the tool `ActivAlignment`. Please consult the [User's Manual for ActivAlignment](#) for more information.

If the lines of text do not appear more or less horizontally in the image, you should use a rotated rectangle or an ellipse as ROI, i.e., a shape for which the orientation can be defined. This orientation is used as the initial orientation of the lines of text to be read.

Extraction of Symbols (Segmentation)

Within each ROI of AVTOCR, the symbols are extracted automatically. You only need to adjust some parameters to tune this segmentation. You may also choose between different segmentation methods, according to the background and to the appearance of the text in the image.

You must set the parameters such that all the symbols that you want to read are segmented correctly in a set of images that is typical for your application. This is very important, because the extracted symbols are the input data for the reading process, which actually transforms the extracted symbols into meaningful characters, which are described by a symbol name. That means, if some parts of the text were not extracted by the segmentation process, they can not be read.

Finding segmentation parameters that allow a robust extraction of all symbols in varying images is usually the most demanding part of the configuration.

OCR Fonts

The extracted symbols are then read using an OCR font.

“Normal” fonts, which, in the following, are denoted as Windows fonts describe the appearance of characters in order to print them. OCR fonts describe the appearance of the characters in order to read (recognize) them, i.e., they contain information about the features that differentiate the individual characters. Basically, an OCR font can be seen as a classifier that is used to assign a logical name, i.e., a semantic meaning to any extracted symbol.

You can choose from the list of ready-to-use OCR fonts, or create your own OCR font. To create your own OCR font, you must train it using a trainfile containing samples of the symbols to read.

You can teach samples manually, or you can generate artificial samples from an installed Windows font. Furthermore, you can derive new samples from existing samples by applying artificial distortions. However, you should be aware that the best results are achieved if the OCR font is trained using real data from the target application.

You must use an OCR font that is suitable for the text to be read. For example, if you want to read text that was printed using the Windows font Arial, you must use an OCR font that has been trained based on samples showing characters in a Windows font that is at least similar to Arial.

Results

The results of ActivOCR are characters, i.e., the read symbols. Furthermore, ActivOCR lets you output additional information (also called *features*), e.g., the confidence with which a symbol could be read, the symbol alternatives, i.e., the next likely readings of the symbols, and the position of the symbols within the words and lines of text.

The results of ActivOCR can be further processed using ActivWordProcess and/or ActivDecision, before you output them via ActivFile, ActivSerial, or ActivDigitalIO.

ActivWordProcess

ActivWordProcess can be used to improve or check the results of ActivOCR based on expressions and dictionaries. You can formulate expressions that must be fulfilled by the results of ActivOCR. If these expressions are not fulfilled, ActivWordProcess tries to replace each symbol that does not fit by one of its alternatives. If dictionaries are used, words that are not contained in the dictionary are replaced by the word out of the dictionary that is most similar to the given word.

As a side effect, ActivWordProcess groups the characters returned by ActivOCR to words and returns them as strings.

Programming Interface

All results of ActivOCR and ActivWordProcess can be accessed via the programming interface (see [section 4.3](#) on page 62). This includes the intermediate results of the segmentation process. With this, it is even possible to import externally generated images and regions into the segmentation process (see [section 4.4](#) on page 70).

1.2 The Sub-Tools of ActivOCR

Besides its *master tool*, ActivOCR provides 8 *support tools*. In [figure 1.1](#), [figure 1.2](#) on page 6, and [figure 1.3](#) on page 7 they are depicted together with other ActivVisionTools that you will use in a typical ActivOCR application.



AVTOCR is the *master tool* of ActivOCR. How to use AVTOCR is described in more detail in [chapter 2](#) on page 11.



AVTOCRView is a *support tool* of ActivOCR. You can use it to inspect the intermediate steps of the symbol extraction. How to use AVTOCRView is described in [section 2.2.4](#) on page 20.



AVTOCRSettings is a *support tool* of ActivOCR. You can use it to set which (intermediate) results are displayed. How to use AVTOCRSettings is described in [section 2.4](#) on page 38.



AVTOCRAdvancedSettings is a *support tool* of ActivOCR. You can use it to tune the symbol extraction. How to use AVTOCRAdvancedSettings is described in [section 2.2.3](#) on page 18.



AVTOCRTrain is a *support tool* of ActivOCR. You can use it to manage the OCR font. This includes the generation and editing of OCR trainfiles as well as the training of the classifiers. How to use AVTOCRTrain is described in [section 2.3](#) on page 22.



AVTOCRTeach is a *support tool* of ActivOCR. You can use it to add example images for symbols to the OCR trainfile. How to use AVTOCRTeach is described in [section 2.3.4](#) on page 28.



AVTOCRGenerate is a *support tool* of ActivOCR. You can use it to generate example images for symbols from any Windows font installed on your PC and to add these example images to the OCR trainfile. How to use AVTOCRGenerate is described in [section 2.3.3](#) on page 26.

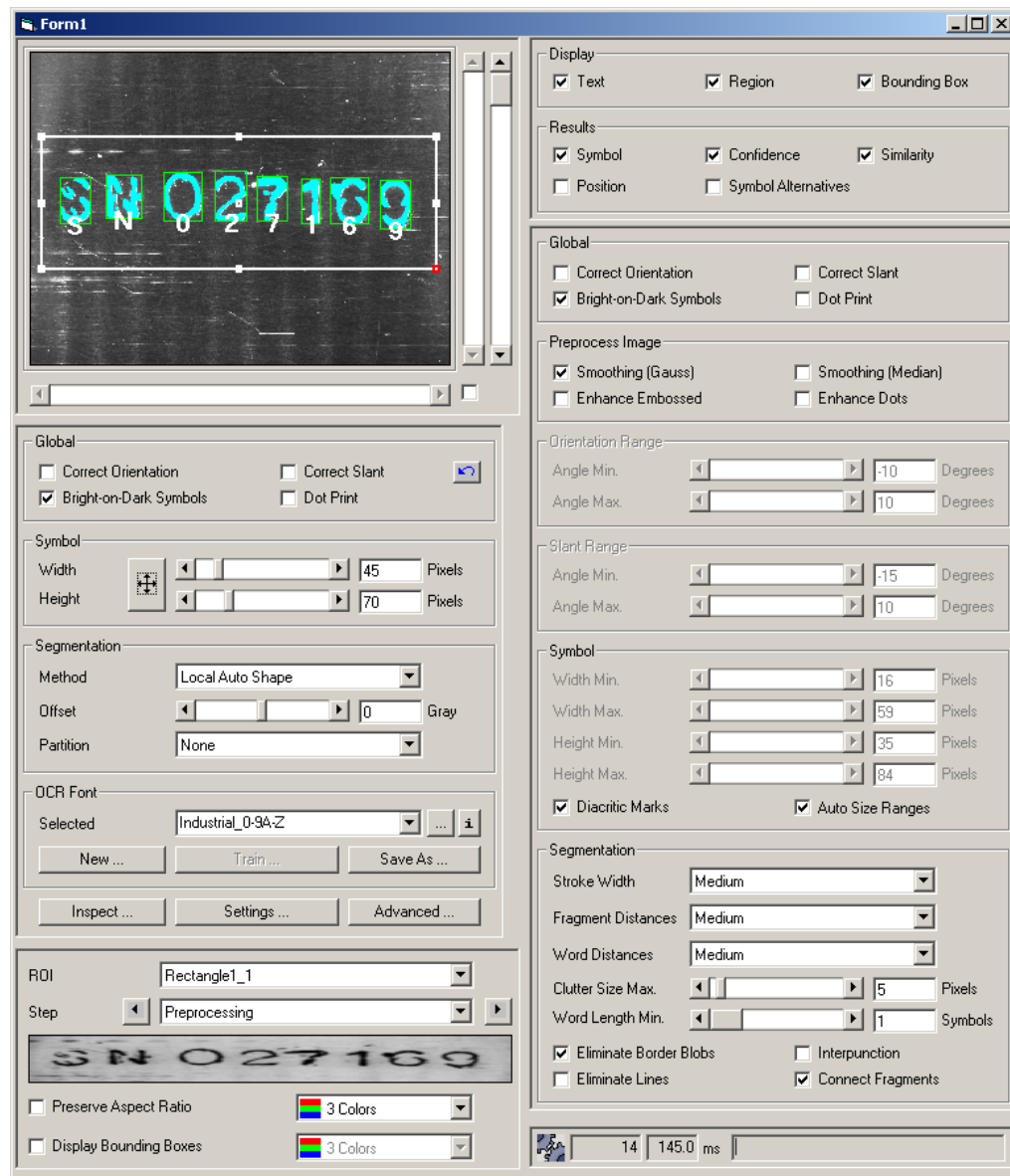


Figure 1.1: The sub-tools of ActivOCR that are primarily used to read characters from images together with suitable other tools.

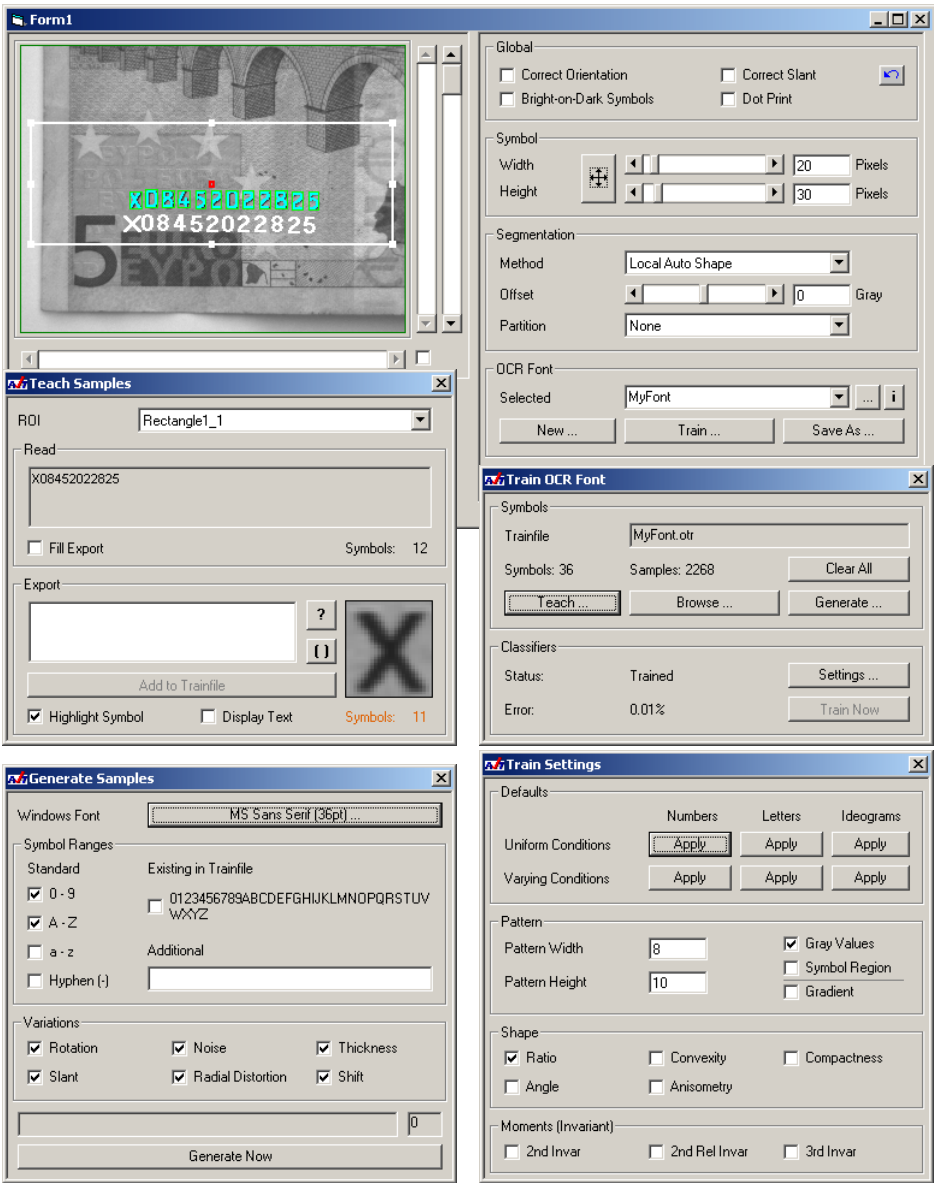


Figure 1.2: The sub-tools of ActivOCR that are primarily used to handle OCR fonts together with suitable other tools.

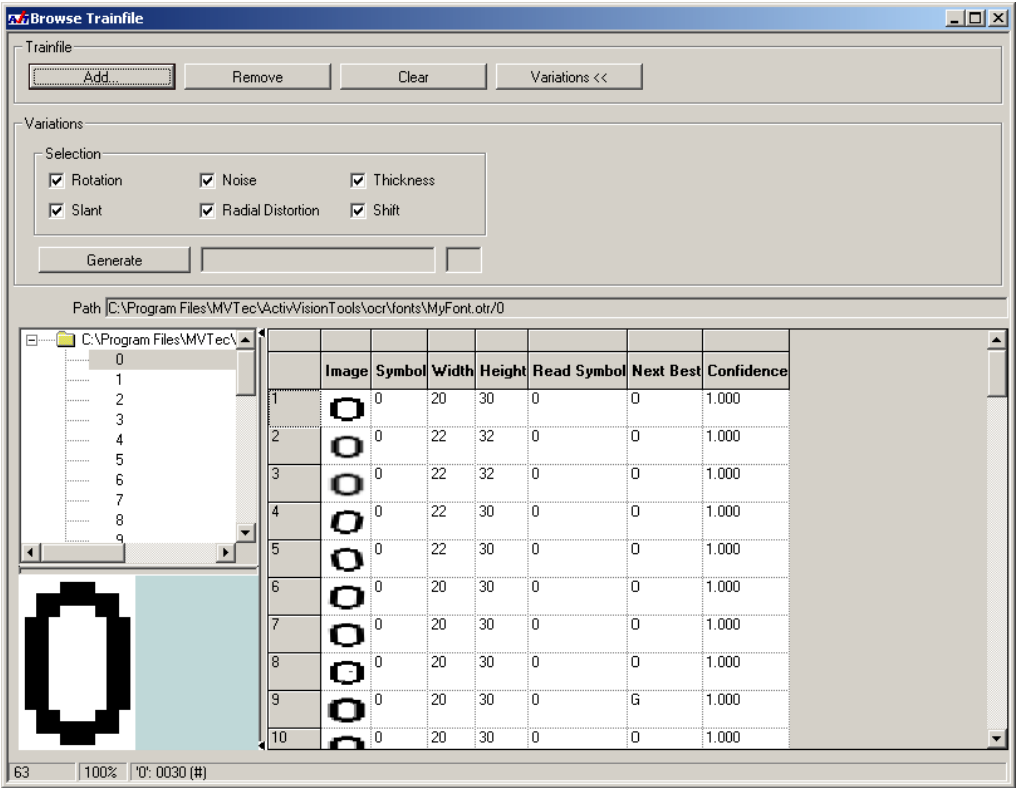


Figure 1.3: The sub-tool AVTOCRTrainFileBrowser .

AVTOCRTrainFileBrowser is a *support tool* of ActivOCR. You can use it to browse and edit the OCR train file. How to use AVTOCRTrainFileBrowser is described in [section 2.3.5](#) on page 32 and [section 2.3.6](#) on page 36.



AVTOCRFontSettings is a *support tool* of ActivOCR. You can use it to tune the OCR classifier. How to use AVTOCRFontSettings is described in [section 4.5](#) on page 74.



1.3 The Sub-Tools of ActivWordProcess

Besides its *master tool*, ActivWordProcess provides 2 *support tools*. In figure 1.4 they are depicted together with other ActivVisionTools that you will use in a typical ActivOCR application.

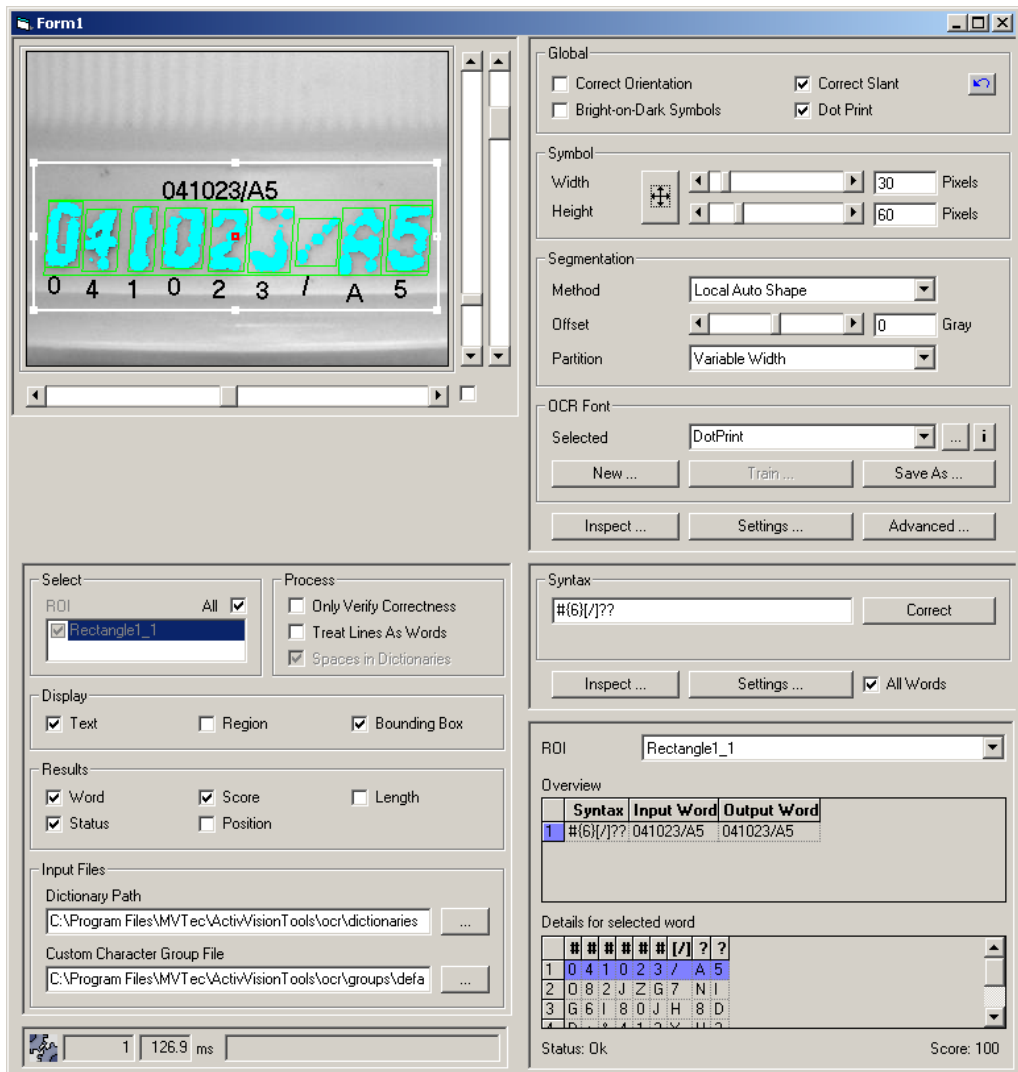


Figure 1.4: The sub-tools of ActivWordProcess together with suitable other tools.

AVTWordProcess is the *master tool* of ActivWordProcess. How to use AVTWordProcess is described in more detail in [section 3.1](#) on page 42.



AVTWordProcessView is a *support tool* of ActivWordProcess. You can use it to inspect the behavior of ActivWordProcess. How to use AVTWordProcessView is described in [section 3.1.2](#) on page 46.



AVTWordProcessSettings is a *support tool* of ActivWordProcess. You can use it to control the behavior of ActivWordProcess. How to use AVTWordProcessSettings is described in [section 3.1.4](#) on page 50.



Chapter 2

Using ActivOCR

This chapter will explain how to use ActivOCR to read characters from images. It describes how to create the search area, how to parameterize the extraction of the symbols, how to provide an appropriate OCR font, and what types of results can be achieved.

The corresponding Visual Basic projects show how to solve an example OCR task.





2.1	Specifying the Search Area	12
2.2	Extracting the Symbols from the Image	14
2.2.1	Setting Basic Parameters	14
2.2.2	Selecting a Segmentation Method	16
2.2.3	Setting Advanced Parameters	18
2.2.4	Inspecting the Effect of the Parameter Settings	20
2.3	Selecting and Creating OCR Fonts	22
2.3.1	The Basic Concept of OCR Fonts	22
2.3.2	Using an Existing OCR Font	24
2.3.3	Generating a New OCR Font From a Windows Font	26
2.3.4	Teaching an OCR Font	28
2.3.5	Editing Training Samples	32
2.3.6	Adding Variations of Existing Samples	36
2.4	Displaying and Selecting Results	38

2.1 Specifying the Search Area Using


ActivOCR lets you choose between various shapes for the regions of interest: axis-aligned rectangles, rotated rectangles, polygons, circles, and ellipses. If you are using a shape that defines a direction, i.e., a rotated rectangle or an ellipse, this direction is used as the default orientation of the lines of text to be read. You specify a region of interest using AVTViewROI, which is a *support tool* of ActivView.

Visual Basic Example

Preparation for the following example:

- ☐ Open the project `rois\ocr_rois.vbp`. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): AVTView () , AVTViewROI () , AVTOCR () , and AVTViewStatus () .
- ☐ Execute the application (Run ▸ Start or via the corresponding button).
- ☐ Open AVTViewFG by clicking on AVTView with the right mouse button and selecting Image Acquisition. Load the image sequence `ocr\money1.seq` via the combo box Input File.

The following steps are visualized in [figure 2.1](#).

- ① First, you have to tell AVTViewROI that it is to create an ROI for ActivOCR by selecting the corresponding entry in the combo box `ActivVisionTool`. In this box, all ROI-processing `ActivVisionTools` are listed that have been placed upon the form. The tools are referenced by the *name* of the corresponding ActiveX control. In the default behavior of Visual Basic, multiple instances of a control are distinguished by a postfix counter, e.g., `AVTOCR1` and `AVTOCR2`. In our example, there is only one item in the combo box, `AVTOCR1`.
- ② Next, select the desired shape of the region of interest.
- ③ To draw the region of interest, move the mouse in the image while keeping the left mouse button pressed. Please experiment at this point with the different shapes. The creation of a polygonal ROI () is more complex: By the first mouse movement, the first side of the polygon is created. You can add a new corner point by clicking on the line with the left mouse button; when you drag it by keeping the mouse button pressed, new polygon sides are created, where you can again add new corner points. To delete a corner point, drag it onto a neighboring corner point.
- ④ Now, you can move the ROI by dragging its pick point in the middle. By dragging the outer pick points you modify its shape. Again, please experiment to get familiar with the ROIs. To finely position an ROI, you can zoom the image and move the displayed

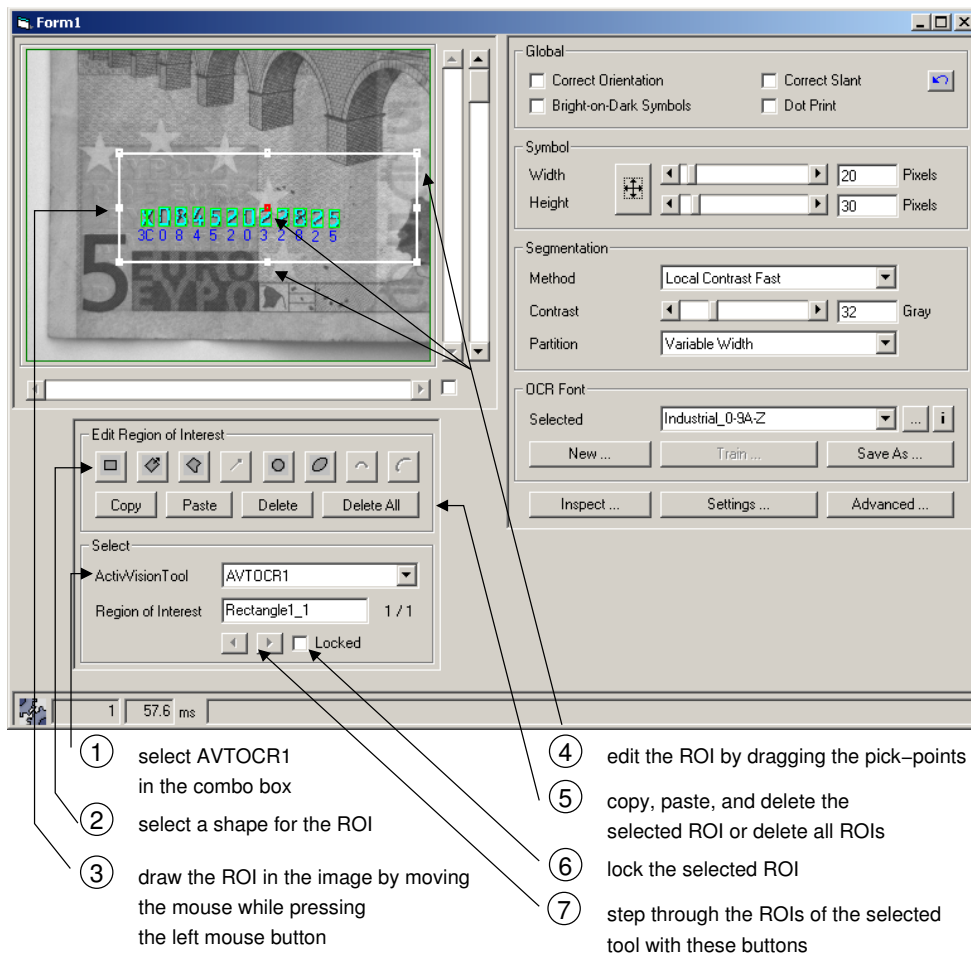




Figure 2.1: Creating the region of interest (ROI).

part using the scrollbars to the right and below the image.

- ⑤ To create a second ROI, repeat step ②. You can also copy, paste, and delete the selected ROI or delete all ROIs.
- ⑥ By checking ☒ Locked you can lock the selected ROI and thus prevent it from accidental editing.
- ⑦ ROIs can be selected by clicking next to it in the image. Alternatively, use the two arrow buttons   to step through all the ROIs of the selected tool.

2.2 Extracting the Symbols From the Image Using

While you created and modified ROIs, ActivOCR already started to work and extracted and read symbols within the ROIs. We now show how to specify what is to be extracted.

2.2.1 Setting Basic Parameters Using

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, you may remove AVTViewROI.
Otherwise, open the project `segmentation_basic\ocr_segmentation_basic.vbp`
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence `ocr\money1.seq`. If necessary, place an axis-aligned, rectangular ROI as shown in [figure 2.2](#); AVTViewFG and AVTViewROI can be opened at run time via a right mouse button click on AVTView.

The following steps are visualized in [figure 2.2](#).

- ① First, you have to set some global parameters that describe the appearance of the symbols in the image. Select
 - ☒ **Correct Orientation** if the orientation of the lines of text varies over the images. If the lines of text are not oriented horizontally but constant over the images, a correspondingly oriented rectangle or an ellipse should be chosen as the ROI.
 - ☒ **Correct Slant** if the symbols are slanted.
 - ☒ **Bright-on-Dark Symbols** if the symbols are brighter than the background.
 - ☒ **Dot Print** if dot prints must be read.

Note that if the symbols appear perspectively distorted, you might use image rectification as described in the User's Manual for ActivGeoCalib, [section 4.2](#) on page 26.

- ② Then, the average size of the symbols must be specified approximately. This can be done by defining the Width and Height via the respective sliders or by setting the values directly. Alternatively, you can specify the size interactively by clicking the button to the left of the sliders. A special ROI appears. Place this ROI around the largest symbol. The ROI can be rotated to make it easy to determine the size of rotated symbols. To finely position this ROI, you can zoom the image and move the displayed part using the scrollbars to the right and below the image (see [figure 2.3](#)). To remove the special ROI, click the button again.

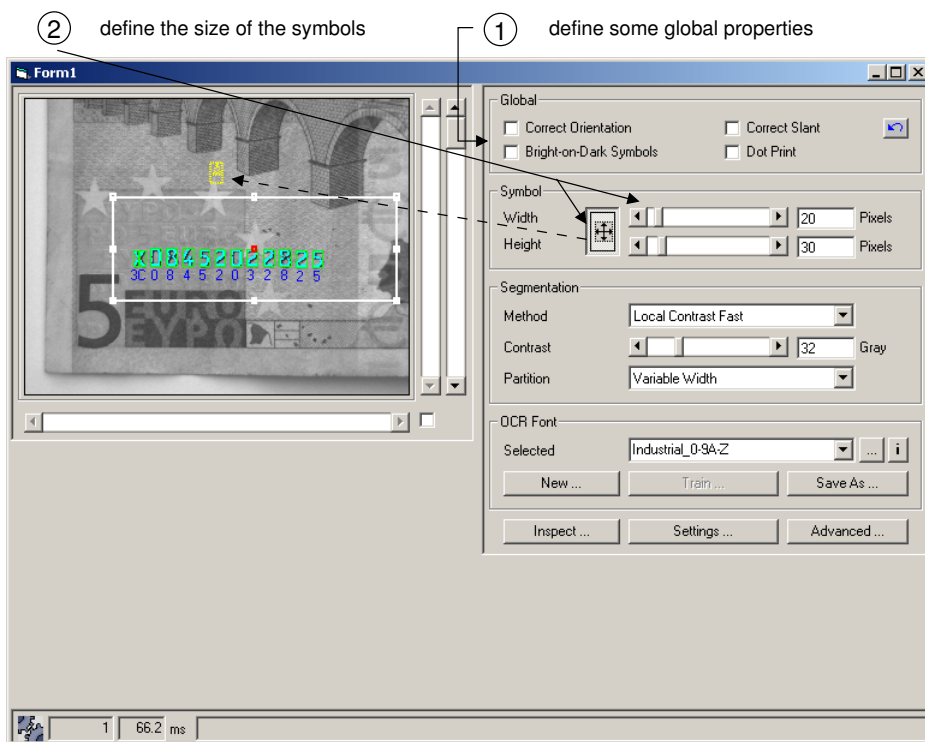


Figure 2.2: Extracting the symbols from the Image.

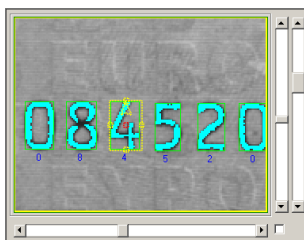


Figure 2.3: Measuring the size of the symbols with the special ROI .

As soon as you modify the average size, the segmentation result changes. If the size is chosen too small, symbols may be split into multiple parts (indicated by multiple characters appearing below the symbol). If the size is too large, multiple symbols may be merged into a single one. We recommend to check in multiple test images whether the size was chosen correctly.

2.2.2 Selecting a Segmentation Method Using

ActivOCR provides powerful segmentation methods, which allow to extract symbols even from difficult backgrounds. The main methods, which already solve many applications, can be found and parameterized in AVTOCR. More advanced methods and parameters can be set via the AVT-OCRAdvancedSettings dialog (see [section 2.2.3](#) on page 18).

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project.
Otherwise, open `segmentation_method\ocr_segmentation_method.vbp`.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.

The following steps are visualized in [figure 2.4](#).

- ① First, select the segmentation method in the combo box Method. Depending on the selected method, you must adjust a parameter (see below).

The available segmentation methods can be grouped into *global* and *local* methods. Generally, if the appearance of symbols and background is homogeneous within the whole ROI, a *global* method might be the right choice. If, in contrast, the appearance varies, e.g., due to a textured background or an inhomogeneous illumination, *local* methods will perform better. Available methods are:

'Global Fixed'	All pixels are divided into "symbol" and "background" based on the given global Threshold.
'Global Auto Brightness'	If the image is dominated by only two gray values (symbols and background), the global threshold can be estimated automatically. The parameter Offset can be used to adjust the threshold.
'Global Auto Shape'	The global threshold is estimated such that the width of the resulting regions equals the selected stroke width (see section 2.2.3 on page 18). The parameter Offset can be used to adjust the threshold.
'Local Contrast Fast'	Symbols that differ locally from the background are extracted. The parameter Contrast defines the minimum contrast between symbols and background.
'Local Contrast Best'	Like 'Local Contrast Fast', additionally, the borders of symbols are enhanced. The parameter Contrast defines the minimum contrast between symbols and background.

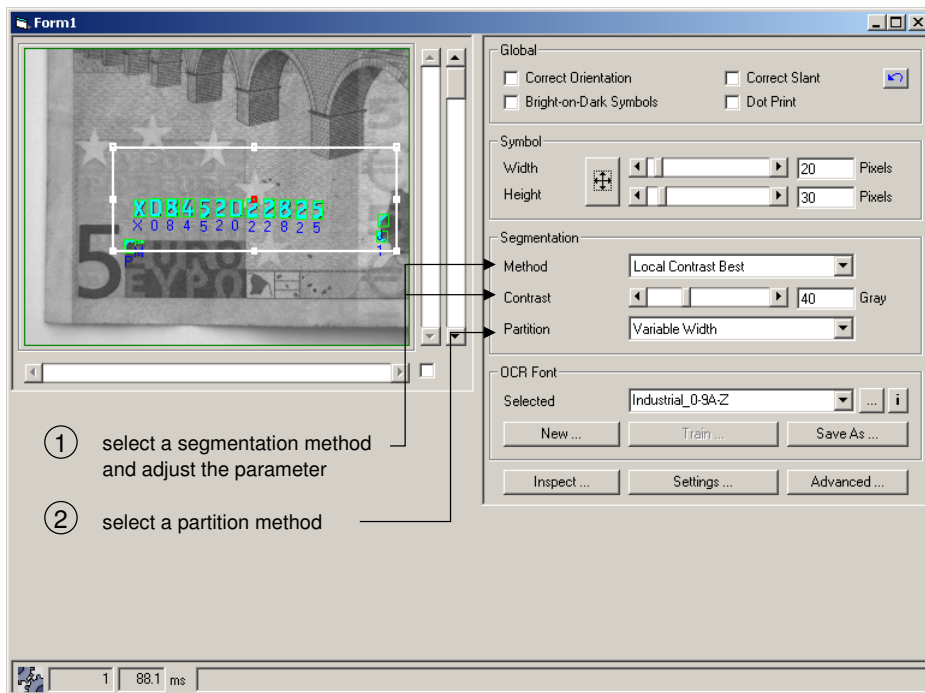


Figure 2.4: Selecting an appropriate segmentation method.

'Local Auto Shape'

The minimum contrast is estimated automatically such that the number of very small regions is reduced. This method is especially suitable for noisy images. The parameter `Offset` can be used to adjust the threshold.

For a detailed description of these methods, refer to [appendix A.1](#) on page 80.

- ② If neighboring symbols are printed close to each other, they may be partly merged. AVT-OCR offers different methods to partition such symbols:


'None'	No partitioning.
'Fixed Width'	The partitioning assumes a constant symbol width.
'Variable Width'	The symbols are partitioned at the position where they have the thinnest connection.
'Fuzzy'	Additionally, the width of the resulting symbols is considered.

For a detailed description of the partition methods, refer to [appendix A.2](#) on page 82.

2.2.3 Setting Advanced Parameters Using

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open `segmentation_advanced\ocr_segmentation_advanced.vbp`.
- ☐ Execute the application (Run ▸ Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ In AVTOCR, press **Advanced** to open the AVTOCRAdvancedSettings dialog. Alternatively, at design time, you can add this dialog by double-clicking on .

The dialog AVTOCRAdvancedSettings is visualized in [figure 2.5](#). The effects of the settings can be inspected using AVTOCRView (see [section 2.2.4](#) on page 20).

- ① The global settings of AVTOCR (see [section 2.2.1](#) on page 14) are also available in AVTOCRAdvancedSettings.
- ② If the image is noisy or the background contains small scale texture, select one of the available smoothing methods (☒ Smoothing (Gauss), ☒ Smoothing (Median)). If you must extract embossed symbols, select ☒ Enhance Embossed; if you want to read dot prints, select ☒ Enhance Dots.
- ③ If you have selected ☒ Correct Orientation in the global settings, you can specify the allowed range for the correction of the orientation.
- ④ If you have selected ☒ Correct Slant in the global settings, you can specify the allowed range for the correction of the slant.
- ⑤ If you uncheck ☒ Auto Size Ranges, you may define the minimum and maximum extent of the symbols directly. If ☒ Auto Size Ranges is selected, these values are determined automatically from the size of the symbols (Width and Height), defined in AVTOCR (see [section 2.2.1](#) on page 14).
- ⑥ If the text in your application contains diacritic marks (e.g., â, é, ö) or punctuation marks (e.g., ,,:;"'"), check the corresponding boxes.
- ⑦ The segmentation contains parameters that describe the appearance of symbols and words. Here, you can specify the stroke width of the symbols (especially important for the segmentation method 'Global Auto Shape', see [section 2.2.2](#) on page 16), the maximum distance between fragments of symbols that shall be merged, and the distance between separate words (used for the determination of the symbols position, see [section 2.4](#) on page 38 and [appendix D](#) on page 109).
- ⑧ If the extracted symbols contain clutter, i.e., small regions near the actual symbols, increase the value of Clutter Size Max. If parts of the symbols are missing, decrease

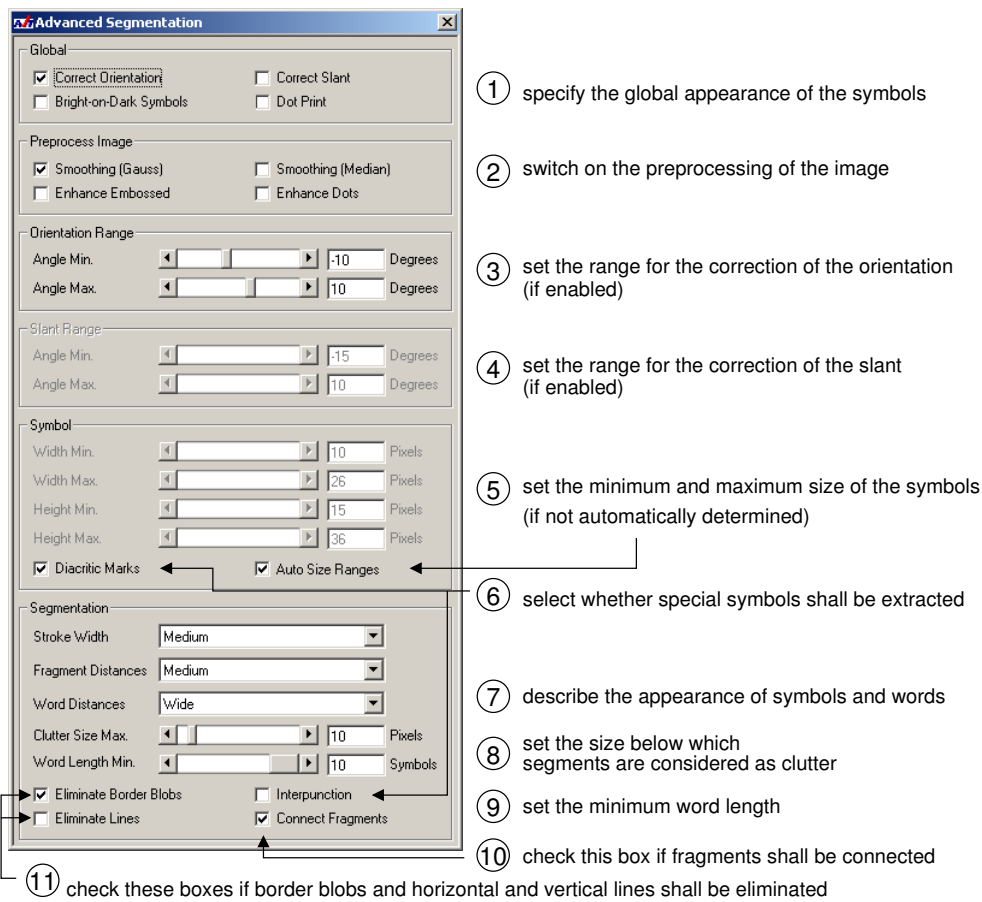


Figure 2.5: Advanced settings for the segmentation of the symbols.


this value.

- ⑨ Define Word Length Min, i.e., the minimum number of symbols that a word must contain. All words that have less than the given number of symbols are rejected. If you want to extract only “long” words, you can use this parameter to suppress incorrectly extracted symbols.
- ⑩ Check ☒ Connect Fragments if the extracted symbols are fragmented, i.e., if a symbol is not extracted as one region but broken up into several parts.
- ⑪ Select ☒ Eliminate Border Blobs to get rid of regions at the border of the ROI. Select ☒ Eliminate Lines if the extraction of the symbols is disturbed by lines that are horizontal or vertical with respect to the lines of text.



2.2.4 Inspecting the Effect of the Parameter Settings Using

Visual Basic Example

Preparation for the following example:

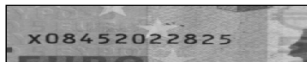
- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open `segmentation_inspection\ocr_segmentation_inspection.vbp`.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ Press **Inspect** to open the AVTOCRView dialog. Alternatively, at design time, you can place this dialog on the form by double-clicking on .

The following steps are visualized in [figure 2.6](#).

- ① Select the ROI whose intermediate results you want to inspect. The corresponding image part will be displayed below.
- ② Select the intermediate result to be inspected either via the combo box or use the two arrow buttons   to step through all the intermediate results.
- ③ You can change the color scheme to be used for the display of the regions.
- ④ You can force the image part to be displayed in its original aspect ratio.
- ⑤ You can display the bounding boxes of the regions in a distinct color scheme.

In the following, the relation between the intermediate results and the individual parameters is described. For a more detailed description of the effect of the parameters, see [appendix A.3](#) on page 85.

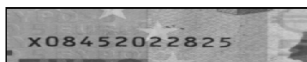
'Preprocessing'



Shows the effect of the preprocessing steps

- ☒ Smoothing (Gauss), ☒ Smoothing (Median),
- ☒ Enhance Embossed, and ☒ Enhance Dots as well as the effect of the selection of
- ☒ Bright-on-Dark Symbols.

'Correct Orientation'



Shows the effect of the selection of

- ☒ Correct Orientation.

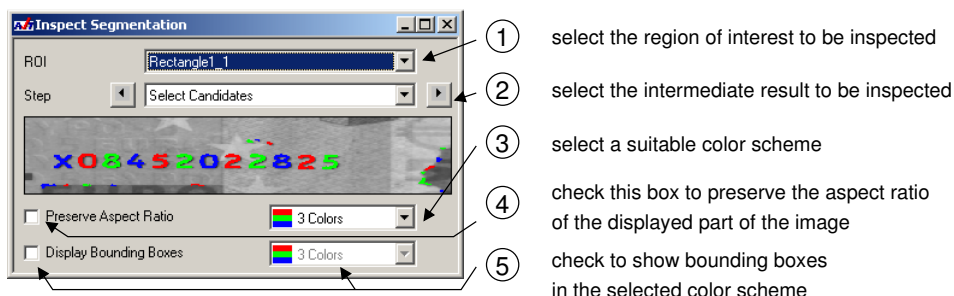
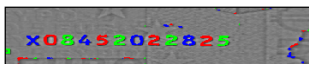


Figure 2.6: Inspecting the intermediate segmentation results.

'Extract Foreground'



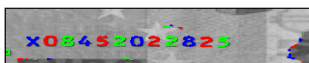
Shows the effect of the selected segmentation method and its parameterization as well as the effect of the selection of ☒ Eliminate Lines and the setting of Stroke Width or Dot Size, respectively. Note that the displayed image itself shows the effects of compensating for local contrast and elimination of lines. However, the original image will be used for subsequent steps.

'Correct Slant'



Shows the effect of the selection of ☒ Correct Slant.

'Select Candidates'



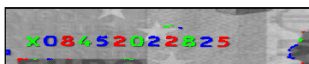
Shows the effect of the setting of the symbol size (Width and Height or Width Min, Width Max, Height Min, and Height Max if ☒ Auto Size Ranges has been unchecked) and of Clutter Size Max.

'Partition Symbols'



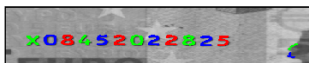
Shows the effect of the selected partition method.

'Connect Fragments'



Shows the effect of the selection of ☒ Connect Fragments and ☒ Dot Print as well as of the setting of Fragment Distances or Dot Distances, respectively.

'Select Symbols'



Shows the effect of the selection of ☒ Diacritic Marks and ☒ Interpunction

2.3 Selecting and Creating OCR Fonts Using

2.3.1 The Basic Concept of OCR Fonts


OCR fonts describe the appearance of the characters in order to read (recognize) them, i.e., they contain information about the features that differentiate the individual characters. Basically, an OCR font can be seen as a classifier that is used to assign a logical name, i.e., a semantic meaning to any extracted symbol. This classifier must be trained. Actually, the OCR font comprises two different classifiers, a box classifier as well as a classifier based on a multilayer perceptron (MLP) and, optionally, the trainfile, which contains the training data of the two classifiers.

ActivOCR comes with a set of ready-to-use OCR fonts, which are already trained (see [section 2.3.2](#) on page 24).

In addition, you may use your own OCR font. To train the OCR font, you need a trainfile, which contains samples for the symbols, i.e., images of the symbols.


ActivOCR provides a lot of support tools to handle OCR fonts. In the following, a short overview is given. The steps are visualized in [figure 2.7](#).

You can either

- ① select an existing OCR font via the combo box **Selected** or the file selector () or
- ② create a new OCR font, which contains an empty trainfile by pressing **New**.

If you have selected an existing OCR font, ActivOCR immediately reads the extracted symbols using the selected OCR font. If the results are satisfying, no further action is necessary.

To improve the selected OCR font (which is only possible, if it contains a trainfile) or to train the new OCR font, you can

- ③ modify its trainfile. Press **Train** to open the dialog AVTOCRTrain, which, alternatively, can be placed at design time by double-clicking on .



Note that changes of the trainfile cannot be undone. If you need to preserve the current state of the trainfile, make a copy of the .otr-file before modifying the trainfile (see [appendix F.1](#) on page 118 for details).

You can add new samples to the trainfile by

- ④ generating them from any Windows font that is installed on your PC (**Generate**; see [section 2.3.3](#) on page 26),
- ⑤ teaching them from images (**Teach**; see [section 2.3.4](#) on page 28), or
- ⑥ varying existing samples (**Browse**; see [section 2.3.6](#) on page 36).

It is possible to browse and edit trainfiles (⑥ in [figure 2.7](#)), e.g., to delete selected samples or to move them from one symbol to another (**Browse**; see [section 2.3.5](#) on page 32).

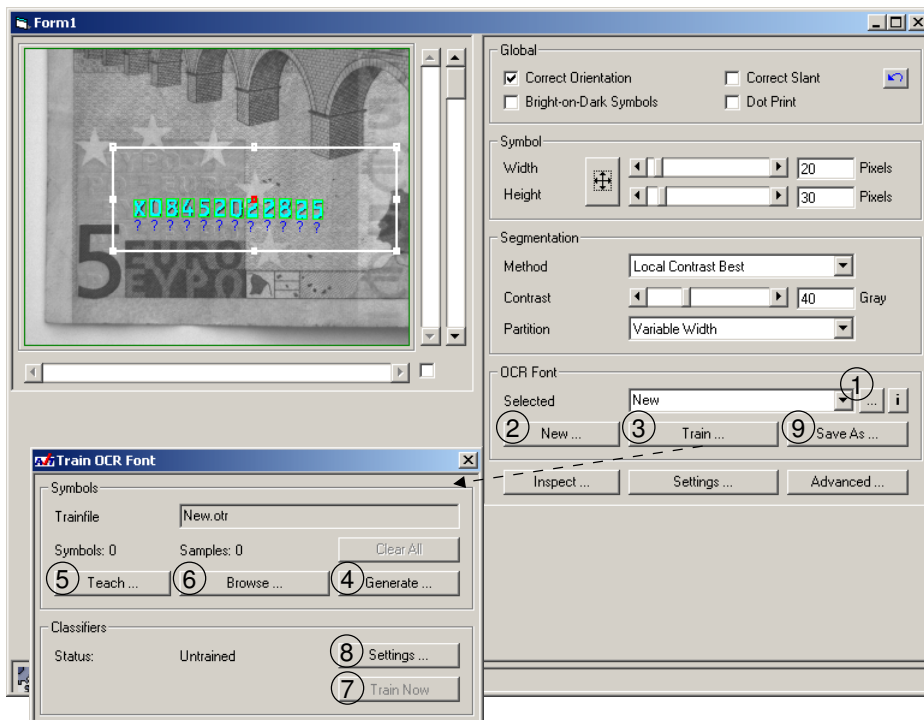


Figure 2.7: Overview over the basic dialog for the handling of OCR fonts .

The modification of the trainfile alone has no effect on the result of the reading process, unless the classifier has been trained using the modified trainfile.

- ⑦ To train the classifier, you must press the button **Train Now**. Depending on the number of samples in the trainfile, this may take a while. The button **Train Now** is only active if the trainfile contains samples that have not been used for the training so far, or if the settings for the OCR font (see below) have been modified.

The result of the training depends on the settings for the current OCR font. Optionally, you can

- ⑧ modify these settings (**Settings**; see [section 4.5](#) on page 74).

The OCR font will be saved automatically upon stopping the application. Alternatively, you can

- ⑨ export the OCR font by clicking **Save As** in the AVTOCR dialog. Note that the trainfile will be part of the exported OCR font. This means that this new OCR font can be modified. If you want to create a read-only copy of an OCR font, copy all the relevant files but the .otr-file directly (see [appendix F.1](#) on page 118 for details).

2.3.2 Using an Existing OCR Font



ActivOCR provides you with a set of ready-to-use OCR fonts, which already solve many applications. Note that these fonts don't come with trainfiles; thus, you cannot modify them.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project.
Otherwise, open the project `font_existing\ocr_font_existing.vbp`.
- ☐ Execute the application (Run ▷ Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.

The following steps are visualized in [figure 2.8](#).

- ① You can select one of the ready-to-use OCR fonts in the combo box or open a file selection box by pressing the button next to it (.
- ② Click on  to display information about
 - the files,
 - the contained symbols,
 - the number of samples used for training, and
 - the error reported after training the classifier
 of the selected font.

ActivOCR comes with the following ready-to-use OCR fonts.

- Document, Document_0-9, Document_0-9A-Z
- DotPrint, DotPrint_0-9, DotPrint_0-9+, DotPrint_0-9A-Z
- HandWritten_0-9
- Industrial, Industrial_0-9, Industrial_0-9+, Industrial_0-9A-Z
- MICR
- OCRA, OCRA_0-9, OCRA_0-9A-Z
- OCRB, OCRB_0-9, OCRB_0-9A-Z
- Pharma, Pharma_0-9, Pharma_0-9+, Pharma_0-9A-Z

For a detailed description of these OCR fonts, see [appendix B](#) on page 93.

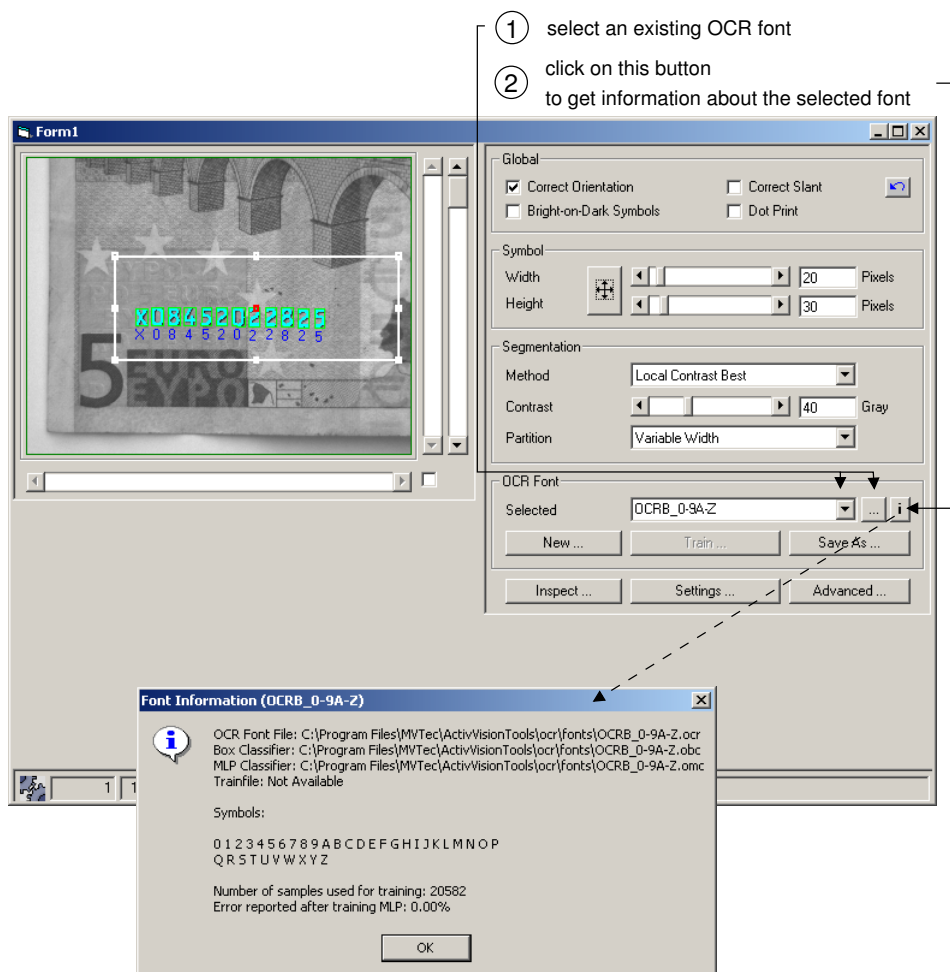




Figure 2.8: Selecting a ready-to-use OCR font .

2.3.3 Generating a New OCR Font From a Windows Font Using and

With AVTOCRGenerate, you can generate samples for the symbols based on Windows fonts. The process is also referred to as font import.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open the project `font_generating\ocr_font_generating.vbp`.
- ☐ At design time, place the AVTOCRTrain dialog on the form by double-clicking on . Alternatively, press `Train` in AVTOCR to open the AVTOCRTrain dialog.
- ☐ Execute the application (`Run > Start` or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ In AVTOCRTrain, press `Generate` to open the AVTOCRGenerate dialog. Alternatively, at design time, you can place this dialog on the form by double-clicking on .



Note that changes of the trainfile cannot be undone. If you need to preserve the current state of the trainfile, make a copy of the .otr-file before modifying the trainfile (see [appendix F.1](#) on page 118 for details).

The following steps are visualized in [figure 2.9](#).

- ① Create a new OCR font.
- ② Select a Windows font from which the samples will be derived.
- ③ In the frame `Symbol Ranges`, you can select the symbols for which samples must be generated. You can select from the predefined ranges and specify additional symbols. If the trainfile already contains some symbols, they can be selected as well. This is useful if an OCR font must be created that is able to read a set of symbols printed in different fonts.
- ④ In the frame `Variations`, you can select the type of variations that shall be applied to the samples. The variation of the samples is used to increase the number of different samples for each symbol. In general, we recommend to use as many samples as possible. So if you are in doubt if a variation type applies or not, you should select it. You can control the amount of variation via the respective properties of AVTOCR.
- ⑤ Then, you can start the generation of the samples by pressing `Generate Now`.
- ⑥ Finally, you start the training of the OCR font by pressing `Train Now` in AVTOCRTrain.

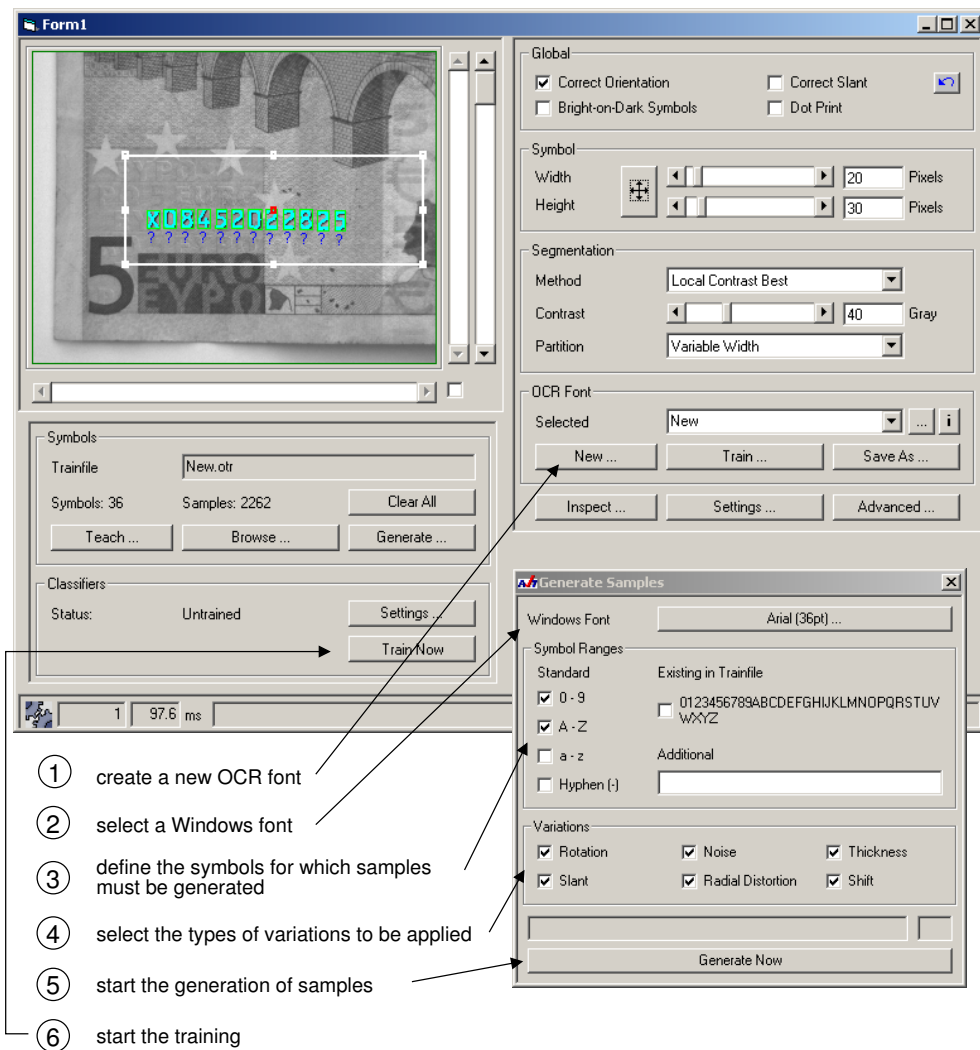


Figure 2.9: Generating samples for an OCR font .

Again, the result of the training depends on the settings for the current OCR font. See [section 4.5](#) on page 74 for details.

Figure 2.9 shows the state of the application between step ⑤ and step ⑥. The trainfile contains already samples for the symbols but the state of the OCR font is still untrained.

2.3.4 Teaching an OCR Font Using


In the previous example (see [section 2.3.3](#) on page 26), a new OCR font has been generated based on the Windows font “Arial”. This Windows font is similar but not identical to the font used to print the serial numbers in the images to inspect. Therefore, using this new OCR font, it can be expected that not all symbols are read correctly.

In fact, if you step through the image sequence (open AVTViewFG by clicking on AVTView with the right mouse button and select Image Acquisition; load the next image of the image sequence by clicking on `Single`) you will detect some erroneously read symbols.

To improve the OCR font or even to create one from scratch, you can add samples extracted from your application images to the trainfile using AVTOCRTeach. Note that only OCR fonts for which an .otr-file is available can be modified.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open the project `font_teaching\ocr_font_teaching.vbp`.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ Press `Teach` to open the AVTOCRTeach dialog. Alternatively, at design time, you can place this dialog on the form by double-clicking on .

The following steps are visualized in [figure 2.10](#).



Note that changes of the trainfile cannot be undone. If you need to preserve the current state of the trainfile, make a copy of the .otr-file before modifying the trainfile (see [appendix F.1](#) on page 118 for details).

- ① Select the ROI that contains the samples to be added to the trainfile.
- ② If the symbols are read already correctly, at least to a large extent, it might be useful to copy the read symbols into the export text box.
- ③ If you have copied the read symbols into the export text box, simply correct the incorrectly read symbols. Otherwise, type in the symbol names of the samples interactively. The currently active sample is displayed in the window right of the export text box. It can also be highlighted in the AVTOCRView (see next double page).

If the ROI contains multiple lines of text, hit `Return` (in C++: `Ctrl` + `Return`) at the end of a line to create a new line in the export text box.

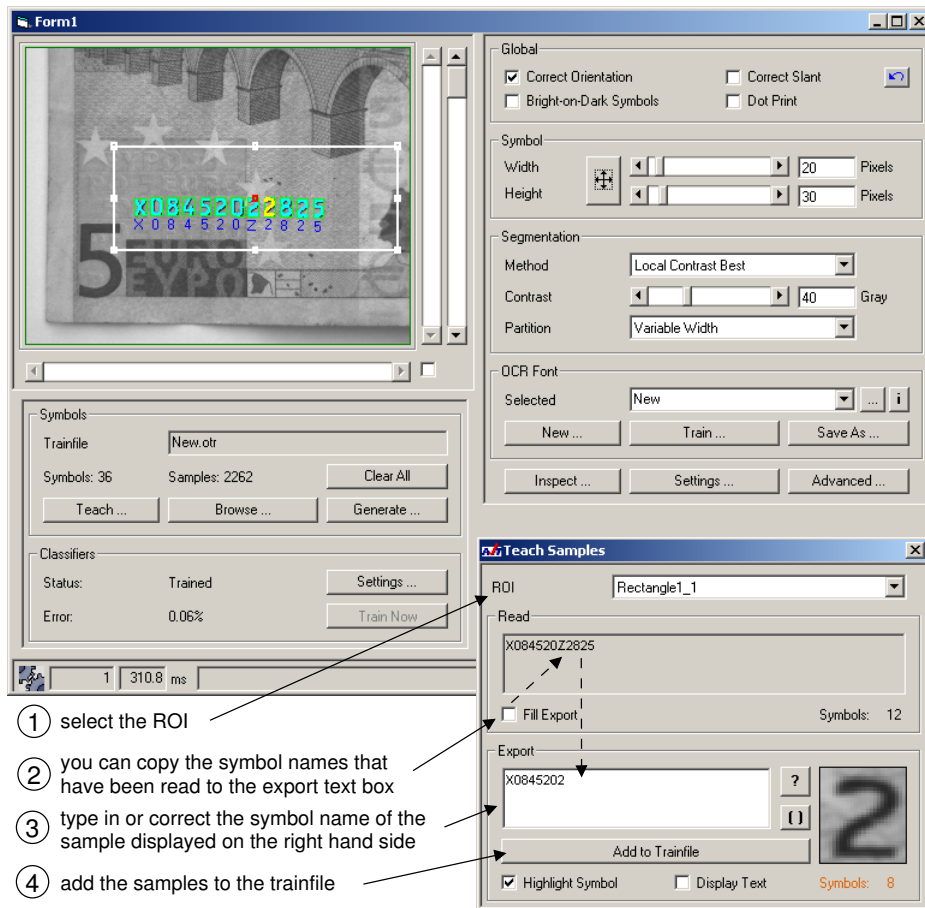


Figure 2.10: Teaching samples for an OCR font .

- ④ After you inserted the symbol names for the samples, add them to the trainfile by pressing **Add to Trainfile**. Then continue with the next image.

We continue with the example on the next double page.

(Teaching, continued)

In the application visualized in [figure 2.11](#), an OCR font has been created based on the Windows font “Times New Roman”. This font contains the characters ‘f’ and ‘i’ as two separate symbols. In the printed document, they are combined into a single symbol, a ligature, for aesthetic appearance. This ligature is extracted as a single symbol and cannot be read correctly, because it is, until now, not known in the OCR font.

To teach simply the ligature, ignore all the other characters as described below and add only the ligature to the trainfile.

The following steps are visualized in [figure 2.11](#).

- ⑤ During the input of the symbol names, it might be necessary to ignore some of the samples, e.g., because you do not need them to be part of the trainfile or in order to skip segmentation errors. To ignore the currently active sample, click on or type + .
- ⑥ To assign symbol names that are longer than one character, e.g., for ligatures, you can click on or type + . This will insert a pair of brackets between which you can enter the symbol name.
- ⑦ There are two possibilities to customize the visualization during the teaching process. You can highlight the currently active symbol and you can display the text you have given as the symbols name directly above its image position in AVTOCRView.
- ⑧ Finally, you must again train the OCR font by pressing . Now, the ligature is read correctly (see [figure 2.12](#)).

Note that only fonts for which .otr-files are available can be taught.

In general, the quality of the classification results increases with the number of different samples per symbol that are used for the training of the classifiers of the OCR font. For this reason, we recommend to use as many samples as possible.

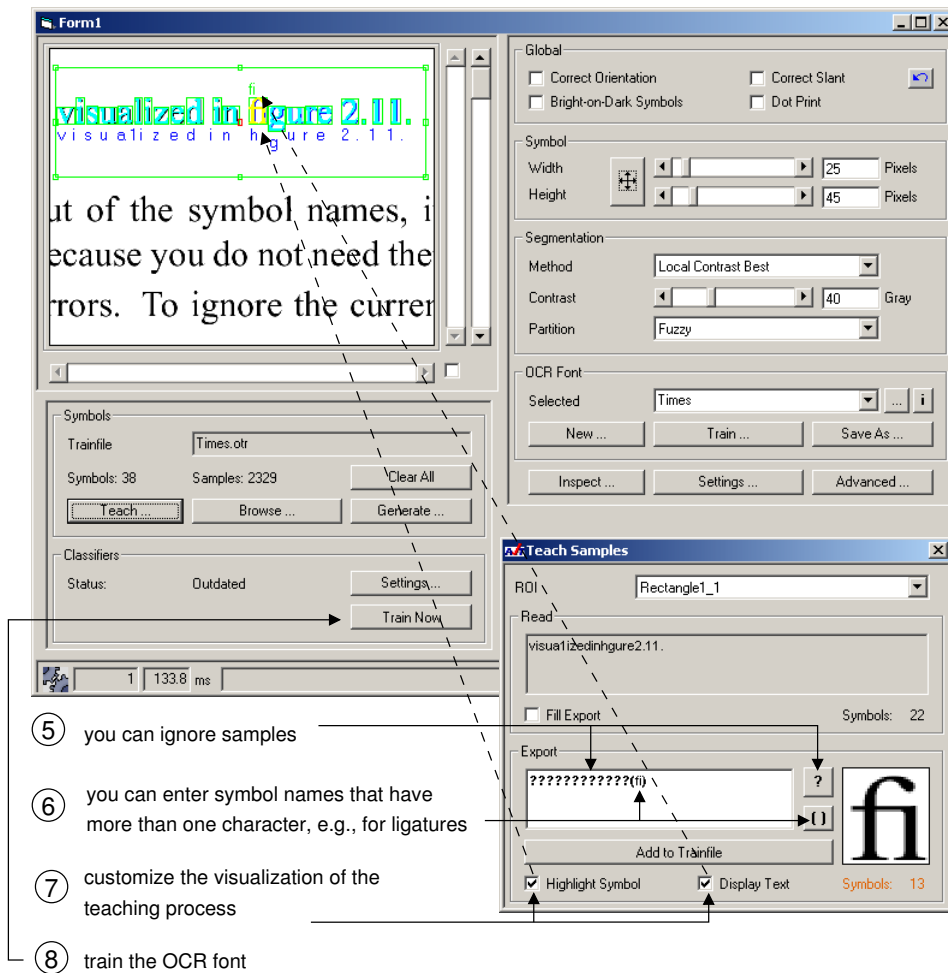


Figure 2.11: Special features of the AVTOCR Teach dialog.

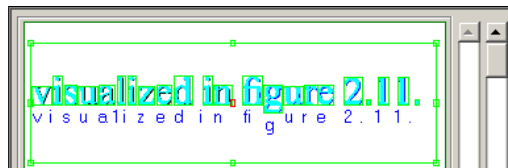



Figure 2.12: Result of reading after the ligature 'fi' has been added to the OCR font .


2.3.5 Editing Training Samples Using

The AVTOCRTrainFileBrowser is a very powerful tool to inspect and modify trainfiles. With it, you can detect and eliminate errors made in the teaching process, e.g., if a wrong symbol was specified for a sample. Note that the ready-to-use OCR fonts provided by ActivOCR don't include trainfiles.

Visual Basic
Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open the project font_editing\ocr_font_editing.vbp.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence ocr\money1.seq.
- ☐ Press **Browse** in AVTOCRTrain to open the AVTOCRTrainFileBrowser dialog. Alternatively, at design time, you can place this dialog on the form by double-clicking on .



Note that changes of the trainfile cannot be undone. If you need to preserve the current state of the trainfile, make a copy of the .otr-file before modifying the trainfile (see [appendix F.1](#) on page 118 for details).

The following steps are visualized in [figure 2.13](#) (only the AVTOCRTrainFileBrowser is shown).

- ① You can inspect the contents of the trainfile, i.e., the individual samples for each symbol. To display the samples of one symbol, select this symbol in the trainfile tree on the left hand side of the AVTOCRTrainFileBrowser. To display all samples of a trainfile, select the respective trainfile.
- ② The samples are displayed in the table on the right hand side of the AVTOCR-TrainFileBrowser. This table provides the following information:

First column	the order number of the sample
Image	the iconic sample
Symbol	the symbol to which the sample is assigned
Width	the width of the iconic sample in pixels
Height	the height of the iconic sample in pixels
Read Symbol	the result of reading the sample with the selected OCR font
Next Best	the second best classification result, according to the currently selected OCR font
Confidence	a measure for the reliability of the read symbol

You can sort the table of samples according to the values of one column by clicking on

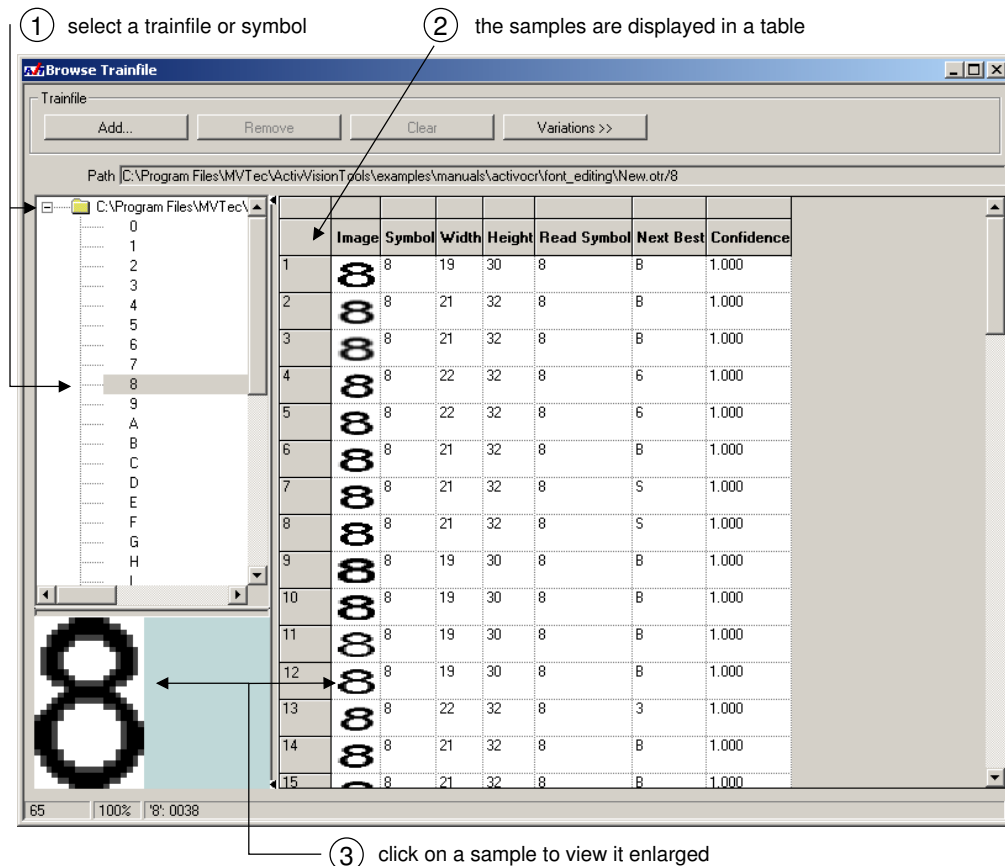


Figure 2.13: Inspect the samples of a trainfile .

the heading of that column. Clicking again on that heading toggles the sorting direction.

If a the value of Read Symbol is not identical to the value of Symbol, the Image of the symbol is marked with a red overlay. This makes it easy to detect samples that do not belong to the selected symbol. Another good method to spot “bad” samples is to sort the table such that samples with low confidences appear on top (click twice on Confidence).

- ③ If you click on a sample, its Image is display enlarged in the lower left corner of the AVTOCRTrainFileBrowser.

We continue with the example on the next double page.

Editing an Available OCR Font, continued

The following steps are visualized in [figure 2.14](#).

- ④ You can add a trainfile to the trainfile tree press **Add**. By default, the trainfile that is currently used by ActivOCR is already loaded in the AVTOCRTrainFileBrowser.
- ⑤ To remove a trainfile from the trainfile tree by pressing **Remove**. This does not change the contents of that trainfile. It is not possible to remove the trainfile that is currently used by ActivOCR.
- ⑥ To delete all samples of the currently selected trainfile press **Clear**;
- ⑦ To move or copy individual samples or groups of samples from one symbol to another, or to delete them, first select the respective samples. To select a sample, left-click on its order number, directly left to the column where the iconic sample is displayed. You can also select multiple consecutive samples by dragging the mouse with the left button pressed over the samples to be selected. It is also possible to achieve this selection by selecting the first sample and then selecting the last sample while pressing the **SHIFT** key. To select non-consecutive samples from the list, select them while pressing the **CTRL** key. Note that drag & drop (see below) does not work for non-consecutive selections.
 - Delete all selected samples by pressing the **DEL** key.
 - Cut and paste selected samples by using the **CTRL**+**X** and **CTRL**+**V** keys. With this, you can move selected samples from one symbol to another, even if the target symbol resides in another trainfile. First select the samples to be moved and press **CTRL**+**X**, then select the target symbol or the target trainfile and press **CTRL**+**V**. If you selected a trainfile as target, the samples will be added to the symbol that has the same name as the source symbol. If this symbol does not exist in the target trainfile, it will be created. If the target symbol resides in the same trainfile as the source symbol, you can move the selected samples also by drag & drop: Click on the image of one of the selected samples and drag the mouse with the left button pressed to the target symbol. Then release the left mouse button.
 - Copy and paste selected samples by using the **CTRL**+**C** and **CTRL**+**V** keys. With this, you can copy selected samples from one symbol to another, even if the target symbol resides in another trainfile. First select the samples to be copied and press **CTRL**+**C** then select the target symbol or the target trainfile and press **CTRL**+**V**. If you selected a trainfile as target, the samples will be added to the symbol that has the same name as the source symbol. If this symbol does not exist in the target trainfile, it will be created. If the target symbol resides in another trainfile as the source symbol, you can copy the selected samples also by drag & drop: Click on the image of one of the selected samples and drag the mouse with

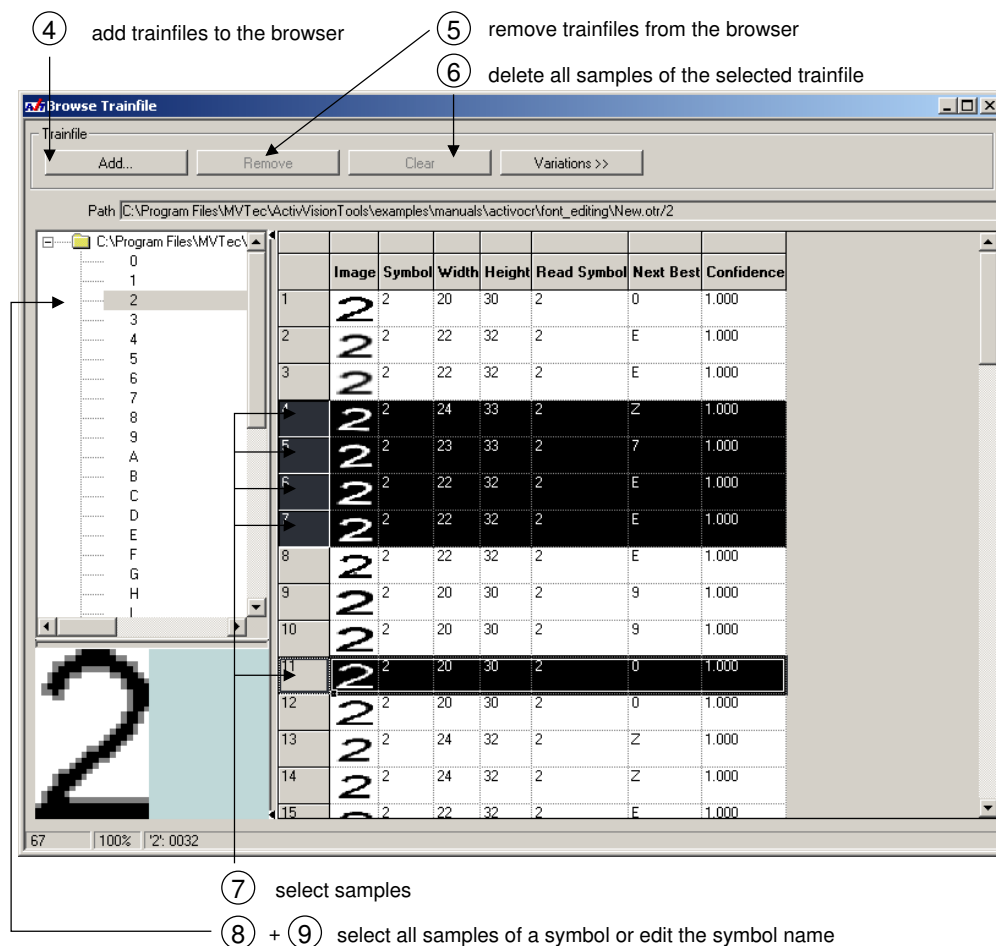


Figure 2.14: Modify the samples of a trainfile .

the left button pressed to the target symbol or trainfile. Then release the left mouse button.

- ⑧ You can move or copy *all* samples of a symbol by selecting the symbol in the trainfile tree, instead of selecting individual samples or groups of samples.
- ⑨ You can edit the symbol name of the selected symbol by again clicking on it and typing in its new name.


To use the modified trainfile, you must train the OCR font by pressing **Train Now** in AVTOCR-Train.

2.3.6 Adding Variations of Existing Samples Using

The best classification results are achieved if the OCR font is trained using real data from the target application. Even though the samples can be derived easily from the images using AVT-OCRTeach (see [section 2.3.4](#) on page 28), this may be time-consuming. To speed up the creation of a large number of different samples per symbol, it is possible to vary existing samples.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. Otherwise, open the project `font_variation\ocr_font_variation.vbp`.
- ☐ Execute the application (Run ▸ Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ Press `Browse` to open the AVTOCRTrainFileBrowser dialog. Alternatively, at design time, you can place this dialog on the form by double-clicking on .



Note that changes of the trainfile cannot be undone. If you need to preserve the current state of the trainfile, make a copy of the .otr-file before modifying the trainfile (see [appendix F.1](#) on page 118 for details).

The following application uses the OCR font that has been taught in [section 2.3.4](#) on page 28. It contains a lot of artificially generated samples (see [section 2.3.3](#) on page 26). For some symbols, additional samples have been added by the teaching process. In the following, variations will be created for one of these real samples.

The following steps are visualized in [figure 2.15](#).

- ① First `Add` a new trainfile, which is used only temporarily.
- ② Move the sample for which variations are to be created to the new trainfile. To move one sample to another trainfile, you must use the `CTRL` + `X` and `CTRL` + `V` keys. If you would copy the sample via drag & drop, it would be contained twofold in the final result.
- ③ Press `Variations` to expand this part of the dialog.
- ④ Select the type of variations that shall be applied to the sample. In general, we recommend to use as many samples as possible. So, if you are in doubt whether a variation type applies or not, you should select it.
- ⑤ Generate the variations by pressing `Generate`. The variations are created for all samples of the selected symbol or trainfile.

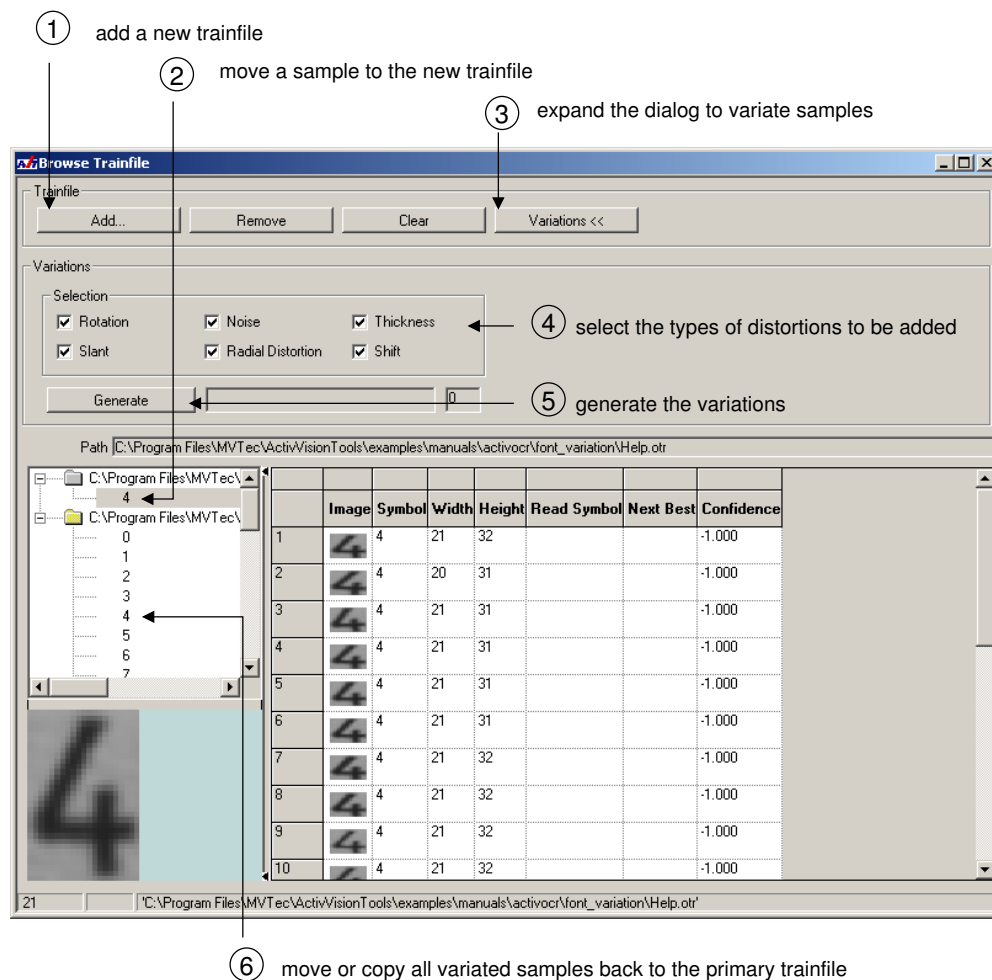


Figure 2.15: Vary the samples of a trainfile .

⑥ Move or copy all samples back to the primary trainfile.

You can also add more than one symbol to the temporary trainfile and select it for the generation of the variations of all contained samples. You can even select individual symbols from the primary trainfile or the whole primary trainfile to generate the variations.

To use the modified trainfile, you must train the OCR font by pressing **Train Now**.



Note that only fonts for which .otr-files are available can be varied.

2.4 Displaying and Selecting Results Using

Using AVTOCRSettings, you can customize the visualization and specify what results (also called *features*) are to be determined. These features are then sent to subsequent ActivVision-Tools, e.g., ActivWordProcess.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTOCRSettings and ActivDataView to the form by double-clicking  and , respectively. You may remove AVTOCRTrain. Execute the application (Run > Start or via the corresponding button).
- ☐ Otherwise, open the project results\ocr_results.vbp. Execute the application (Run > Start or via the corresponding button).
- ☐ Load the image sequence ocr\money1.seq.

The following steps are visualized in [figure 2.16](#).

- ① Customize the visualization of the extraction and reading of symbols by checking the corresponding box in the frame Display. Possible choices are:

<input checked="" type="checkbox"/> Text	display the read symbol names below the extracted symbol
<input checked="" type="checkbox"/> Region	display the extracted symbol
<input checked="" type="checkbox"/> Bounding Box	display a bounding box around each extracted symbol
- ② The result of ActivOCR are the individual symbols. For each symbol, the following features are calculated if you check the corresponding box in the frame Results. Available features are:

<input checked="" type="checkbox"/> Symbol	the read symbol name
<input checked="" type="checkbox"/> Confidence	the confidence measure from the MLP classifier
<input checked="" type="checkbox"/> Similarity	the similarity measure from the box classifier
<input checked="" type="checkbox"/> Position	a code that indicates the position of the symbol with respect to the word and line of text in which it appears (0: within a word; 1: begin of a word; 2: end of a word; 3: a single symbol; 5: begin of a line of text; 10: end of a line of text; 21: first symbol at all; 42: last symbol at all). See appendix D on page 109 for a detailed description of the position code
<input checked="" type="checkbox"/> Symbol Alternatives	a list of the next best alternatives, sorted according to their confidence

Note that the latter two features cannot be unchecked if ActivOCR is connected with

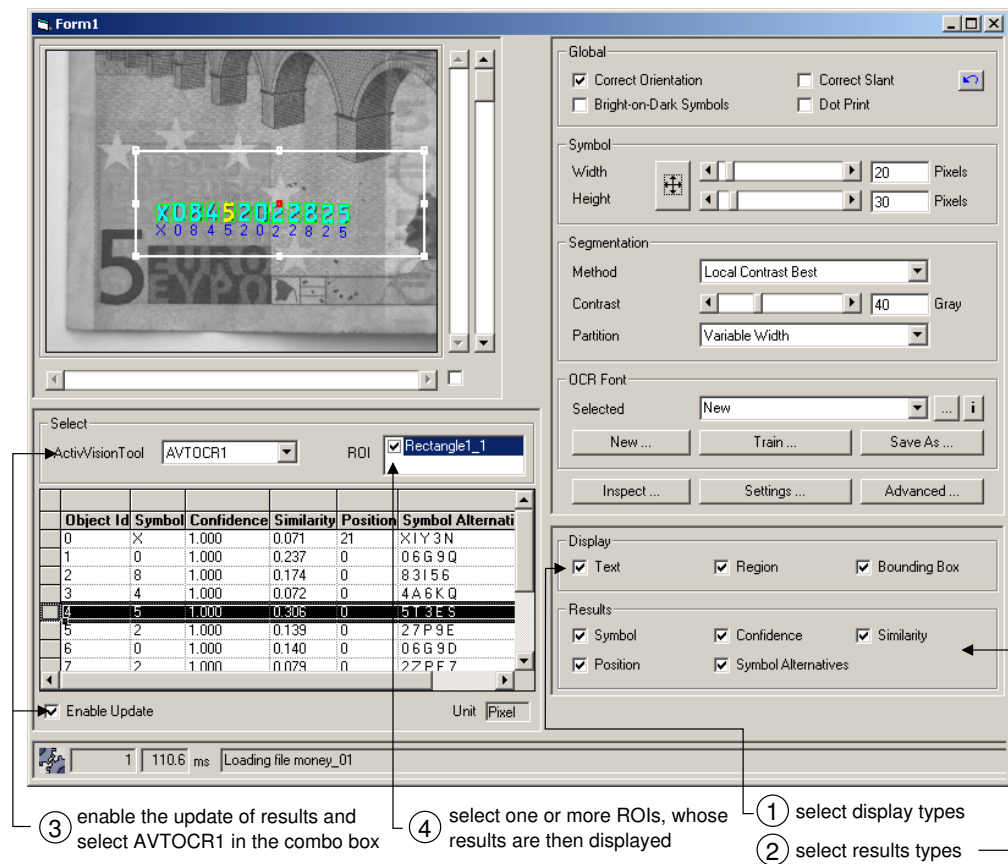


Figure 2.16: Customizing visualization and selecting results.

ActivWordProcess, because ActivWordProcess needs this information to group the symbols to words and to correct them.

- ③ To view the determined features, enable the update of results in AVTDataView and select AVTOCR1 in the combo box ActivVisionTool. If there is only one tool, as in our example, it is selected automatically.
- ④ A list of the ROIs of AVTOCR1 appears. Select the ROIs whose results you want to examine by checking their box. Their results are then displayed in a table (spreadsheet). The columns correspond to the selected features, the rows to the extracted *symbols*.

You can also display the distribution of the values of the numerical features in a histogram using ActivFeatureHistogram, which can be opened by clicking on AVTView with the right mouse button and selecting Feature Histogram in the appearing context menu.

Chapter 3

Combining ActivOCR with other ActivVisionTools

While the previous chapter explained how to read characters from images, this chapter focuses on how to further process, evaluate, and output the results using other ActivVisionTools. Especially, the grouping of the characters into words and their correction is addressed. How to access results and evaluations via the programming interface is described in [chapter 4](#) on page 57.

3.1	Improving and Checking OCR Results Using ActivWordProcess	42
3.1.1	Processing the Words Read by ActivOCR Using ActivWordProcess . .	42
3.1.2	Inspecting the Effect of the ActivWordProcess	46
3.1.3	Verify the Correctness of Words	48
3.1.4	Displaying and Selecting Results	50
3.2	Evaluating Results	52
3.3	Output of Results	54

3.1 Improving and Checking OCR Results Using

ActivWordProcess gives you the possibility to improve or check the OCR results using expressions or dictionaries.




The results of ActivOCR are characters, which are grouped to words based on the position of the symbol regions. With ActivWordProcess, you can compare the words with the entries of a dictionary or with an expression that describes the syntax of the words. If the word is contained in the dictionary or complies with the expression, it is not changed. Otherwise, an attempt is made to find the best correction based on the dictionary or the specified syntax, respectively.

It is also possible, to simply check whether the words comply with the syntax string.

3.1.1 Processing the Words Read by ActivOCR Using

Visual Basic Example

Preparation for the following example:

- ☐ Open the project `wordprocess_basic\ocr_wordprocess_basic.vbp`. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): AVTView () , AVT-OCR () , and AVTWordProcess () .
- ☐ Execute the application (Run ▸ Start or via the corresponding button) and load the image sequence `ocr\money1.seq`. If necessary, place an axis-aligned, rectangular ROI as shown in [figure 3.1](#).
- ☐ Select the OCR font OCRB. Note that the OCR font New, which is the one that has been generated in [section 2.3.3](#) on page 26, is not perfectly suited for the used image sequence. The OCR font OCRB_0-9A-Z would provide better results. In the example, the OCR font New is used intentionally to demonstrate cases where ActivOCR provides erroneous results.

The following steps are visualized in [figure 3.1](#).

In the text box of ActivWordProcess, you can enter a string that describes the syntax of the read words or that specifies a dictionary. A detailed description of the possible components of this string is given on the next double page. In the example, we specify that the read word must start with an uppercase letter (“A”) followed by eleven digits (“#{11}”).

- ① Enter a syntax string that describes the syntax of the word to be read.
- ② ActivWordProcess tries to correct the words it receives from ActivOCR according to the given syntax string. The result is displayed in AVTView, if you didn’t disable this behavior via AVTWordProcessSettings as described further below. In the example, the character is now read correctly.

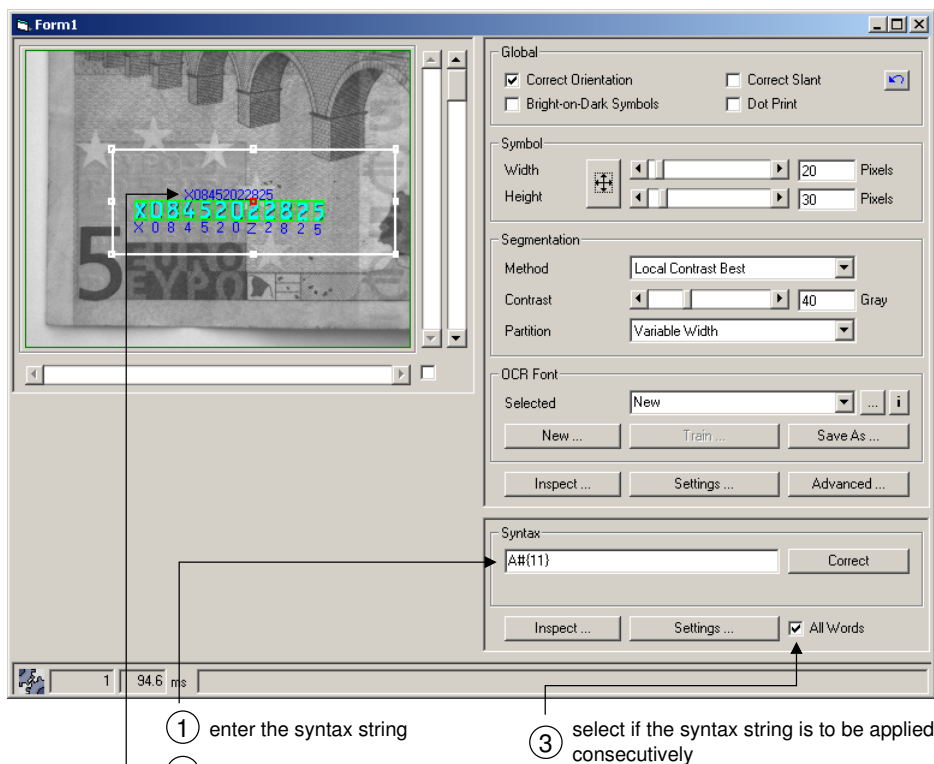


Figure 3.1: The basic usage of ActivWordProcess.

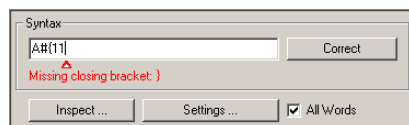


Figure 3.2: Online help of ActivWordProcess during the input of the syntax string.

- ③ The syntax string consists of syntax words separated by spaces. Each syntax word is applied to the appropriate extracted word. If ☒ All Words is checked and more words are found than specified in the syntax string, the syntax string is repeated. If ☒ All Words is unchecked, excess words are not corrected but provided with an error status.

During the input of the syntax string, ActivWordProcess displays information about a possible continuation of the syntax string (see figure 3.2). By pressing the button **Correct**, an incorrect syntax string is corrected. The tooltip of the button shows how ActivWordProcess would correct the string. Note that the resulting syntax string might not represent the expression you intended to enter; it is just a valid syntax string that has been derived from the incorrect input.

(Processing Words, continued)

Description of the Syntax

The syntax words that form the syntax string may either represent a dictionary reference or an expression.

The dictionary reference <my_dictionary> stands for a word that must be contained in the dictionary file my_dictionary.txt.

Each expression consists of at least one of the following atoms, each of which represents a character group:

Atom	Matches
#	0-9
@	A-Za-z
A	A-Z
a	a-z
?	any character
[abc]	any of the characters specified
0-9	any of the characters specified in the Custom Character Group File

The last atom (0-9) refers to a custom character group. It is possible to define up to ten custom character groups.

For each atom, optionally, one of the following modifiers can be given:

Modifier	Effect
*	The previous atom may occur an arbitrary number of times
{n,m}	The previous atom may occur n to m times
{n}	The previous atom must occur exactly n times
!	The previous atom is negated, e.g. [abc]! matches every character except “a”, “b”, or “c”.

To avoid ambiguities, only one occurrence of a dynamic repeat count (* or {n,m}) may occur within a single syntax word.

When specifying characters, you can use ranges as in “[0-9A-Fa-f]”. The literal characters “[”, “]”, “-”, and “\” must be escaped by preceding them with the character “\”.

For a detailed description of the file format of the dictionary and custom character group files, see [appendix E.1](#) on page 114. Examples for syntax strings are given in [appendix E.2](#) on page 115.

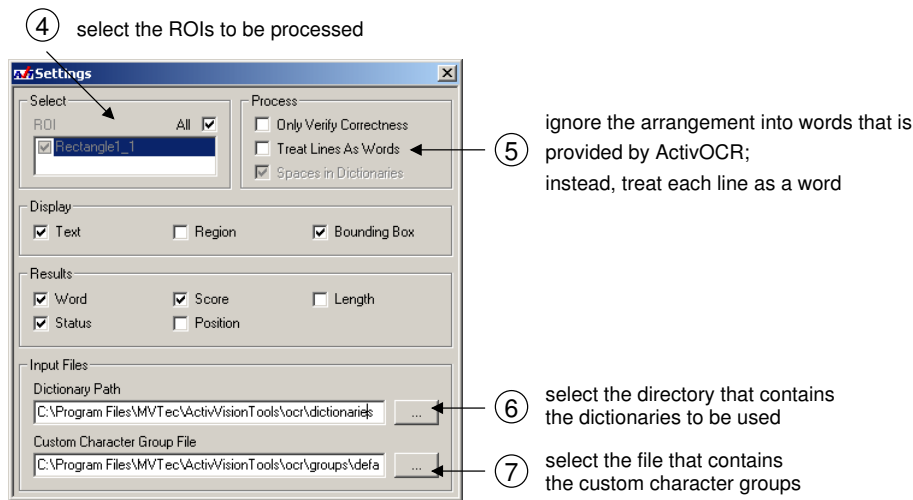





Figure 3.3: Configuring ActivWordProcess.

Further Settings

You can configure ActivWordProcess via its *support tool* `AVTWordProcessSettings`, which can be opened by pressing `Settings` in the `AVTWordProcess` dialog. Alternatively, you can place this tool on the form at design time by double-clicking on .

We continue with the example from the previous double page. The following steps are visualized in [figure 3.3](#).


- ④ If you created more than one ROI for ActivOCR, you can select the ROIs from which the results should be processed by ActivWordProcess. In the example, only one ROI is available.
- ⑤ In some cases, the arrangement of the symbols into words, which is carried out by ActivOCR, might not meet your expectations, e.g., due to highly varying character spacings. In those cases, it is more robust to treat the whole line as a single word. To do so, select ☒ `Treat Lines As Words`.
- ⑥ If you are using a dictionary reference in the syntax string, ActivWordProcess expects the corresponding dictionary file by default in the subdirectory `ocr\dictionaries` of the base directory of your ActivVisionTool installation. You can select another directory by pressing the button  to the right of the Dictionary Path.
- ⑦ Custom character groups are stored in the so-called Custom Character Group File (default: `ocr\groups\default.txt`). You can select your own file by pressing  to the right of the Custom Character Group File.

3.1.2 Inspecting the Effect of the ActivWordProcess Using

ActivWordProcess corrects the results of ActivOCR based on the alternatives for the read characters. You can inspect the correction process using AVTWordProcessView.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project.
Otherwise, open `wordprocess_inspection\ocr_wordprocess_inspection.vbp`.
- ☐ Execute the application (Run ▸ Start or via the corresponding button) and load the image sequence `ocr\money1.seq`.
- ☐ Press **Inspect** in AVTWordProcess to open AVTWordProcessView. Alternatively, at design time, you can place this dialog on the form by double-clicking on .

The following steps are visualized in [figure 3.4](#).

- ① Select the ROI whose results you want to inspect.
- ② A color-coded list of all words read from the selected ROI shows for each word its Syntax, the Input Word, and the, possibly corrected, Output Word. The colors stand for the following status of the correction of the word, which is displayed at the bottom of the dialog:

gray	No correction was necessary (Status: Ok).
green	A correction could be applied successfully (Status: Corrected).
red	A correction was not possible. There are two possible reasons for this behavior: <ol style="list-style-type: none"> 1. Based on the available symbol alternatives, it was not possible to correct the word such that it complies with the given syntax (Status: No Match). 2. The extracted line of text contains more words than the syntax string and <input type="checkbox"/> All Words is not selected (Status: End of Syntax).

The lower part of AVTWordProcessView provides details for the selected word:

- ③ In the heading of the list, the allowed syntax for the selected word is given character by character.
- ④ The list entries show the read symbols in the first row as well as their alternatives in the following rows. The symbol with the highest confidence that complies with the syntax

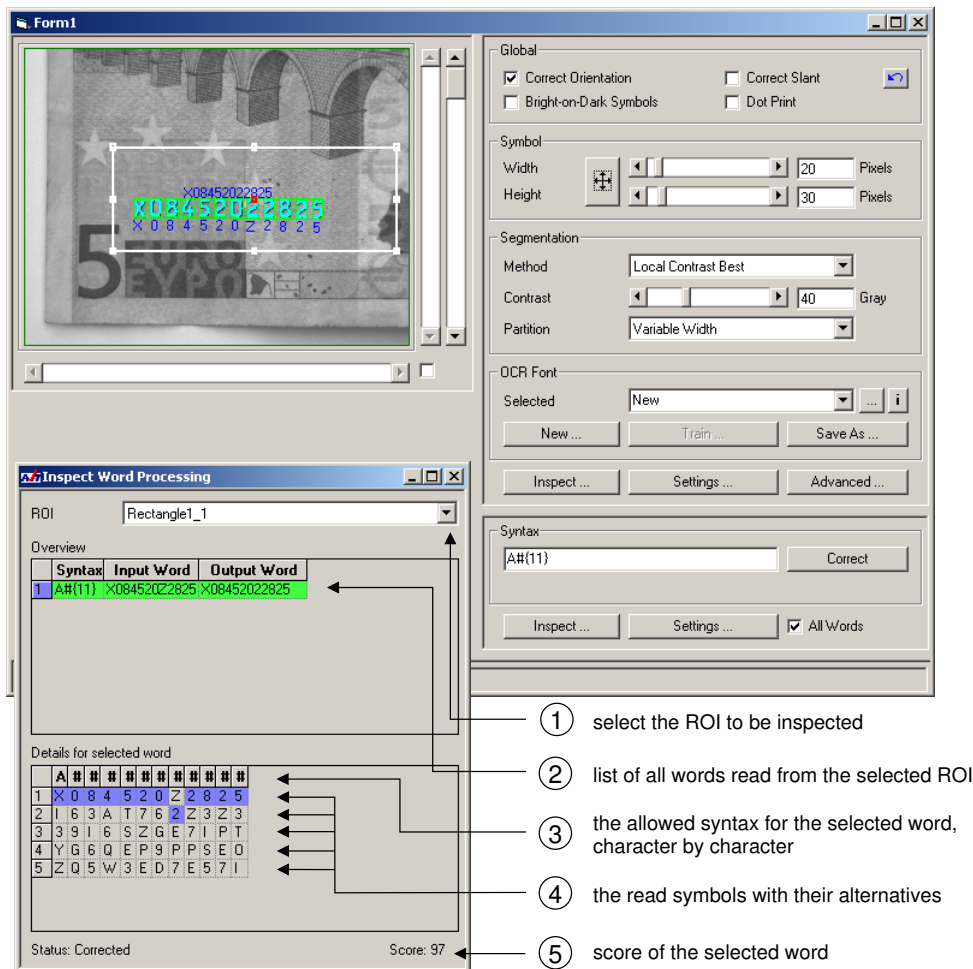


Figure 3.4: Inspection of ActivWordProcess.

is highlighted in blue. If no applicable symbol is available, the respective syntax atom in the heading is highlighted in red.

If the syntax word is a dictionary reference, only the matching word from the dictionary is displayed.



- ⑤ The Score reflects the similarity of the Output Word to the Input Word. It ranges from 0 (no accordance or no correction possible) to 100 (identity).

3.1.3 Verify the Correctness of Words Using

Instead of correcting the results of ActivOCR, you can use ActivWordProcess to “only” verify the correctness of the read words.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using it. Otherwise, open `wordprocess_verification\ocr_wordprocess_verification.vbp`.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence `ocr\money1.seq`; AVTViewFG and AVTViewROI can be opened at run time via a right mouse button click on AVTView.
- ☐ Press **Inspect** to open the AVTWordProcessView dialog and **Settings** to open the AVTWordProcessSettings dialog. Alternatively, at design time, you can place these dialogs on the form by double-clicking on  and .

The following steps are visualized in [figure 3.5](#).

- ① To verify the compliance of the extracted word with the given syntax, check ☒ Only Verify Correctness in AVTWordProcessSettings.
- ② In AVTWordProcessView, a color-coded list of all words read from the selected ROI shows for each word its Syntax, the Input Word, and the Output Word. If ☒ Only Verify Correctness has been selected, the Input Word and the Output Word are identical. The colors stand for the following status of the verification, which is displayed at the bottom of the dialog:

gray	The word complies to the syntax (Status: Ok).
red	The word does not comply to the syntax. There are two possible reasons for this behavior: <ol style="list-style-type: none"> 1. The extracted word does not comply to the given syntax word (Status: No Match). 2. The extracted line of text contains more words than the syntax string and <input checked="" type="checkbox"/> All Words is not selected (Status: End of Syntax).
- ③ The Score is set to 100 if the word is correct and it is set to 0 if the word could not be verified.

The rest of AVTWordProcessView behaves as described in [section 3.1.2](#) on page 46.

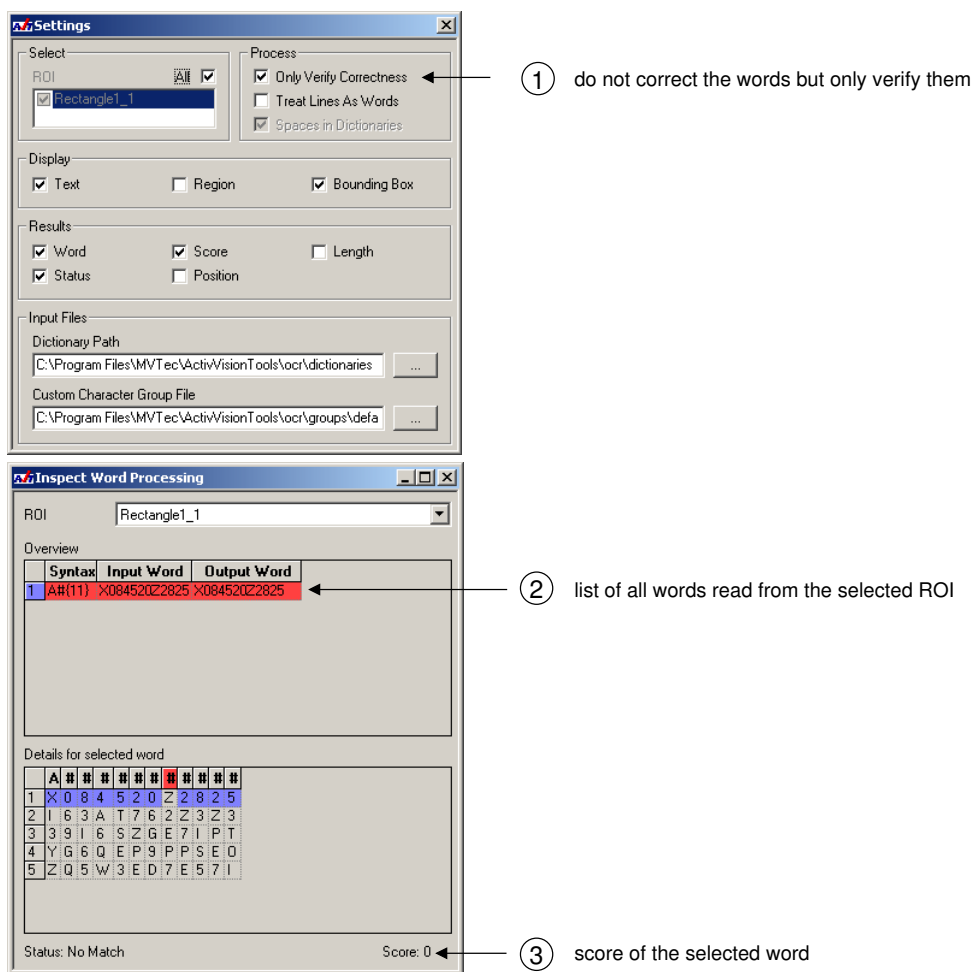



Figure 3.5: Verification of words.

3.1.4 Displaying and Selecting Results Using

Using `AVTWordProcessSettings`, you can customize the visualization and specify what results (also called *features*) are to be determined. These features are then sent to subsequent `ActivVisionTools`, e.g., `ActivDecision`.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add `AVTWordProcessSettings` and `ActivDataView` to the form by double-clicking , respectively. Execute the application (Run ▸ Start or via the corresponding button).
- ☐ Otherwise, open `wordprocess_results\ocr_wordprocess_results.vbp`. Execute the application (Run ▸ Start or via the corresponding button).
- ☐ Load the image sequence `ocr\money1.seq`.

The following steps are visualized in [figure 3.6](#).

- ① Customize the visualization of the extraction and reading of symbols by checking the corresponding box in the frame `Display`. Possible choices are:

<input checked="" type="checkbox"/> Text	display the word below the extracted symbol (or above it, if <input checked="" type="checkbox"/> Text is selected also in <code>AVTOCRSettings</code>)
<input checked="" type="checkbox"/> Region	display the extracted symbols
<input checked="" type="checkbox"/> Bounding Box	display a bounding box around the whole word
- ② Select the features to be determined by checking the corresponding box in the frame `Results`. Available features are:

<input checked="" type="checkbox"/> Word	the corrected or verified word
<input checked="" type="checkbox"/> Score	a measure for the similarity of the Input Word and the Output Word, ranging from 0 (no accordance) to 100 (identity)
<input checked="" type="checkbox"/> Length	the length of the word in characters
<input checked="" type="checkbox"/> Status	the status of the word (Ok, Corrected, No Match, or End of Syntax)
<input checked="" type="checkbox"/> Position	a code that indicates the position of the word with respect to the line of text in which it appears (4: begin of a line of text; 8: end of a line of text; 20: first word at all; 40: last word at all, see appendix D on page 109 for a detailed description of the position code)
- ③ To view the determined features, enable the update of results in `AVTDataView` and select `AVTWordProcess1` in the combo box `ActivVisionTool1`.

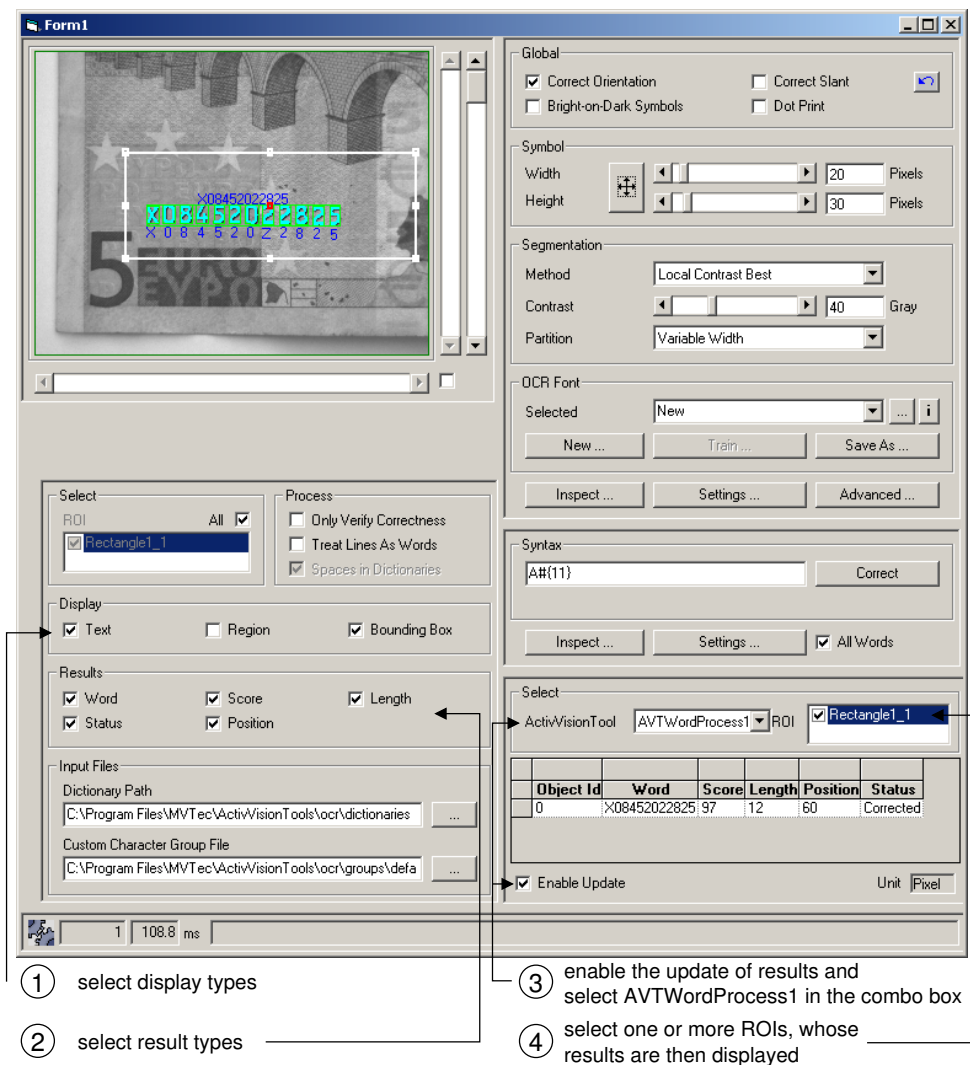


Figure 3.6: Customizing visualization and selecting results.

- ④ Select the ROIs whose results you want to examine by checking their box. Their results are then displayed in a table (spreadsheet). The columns correspond to the selected features, the rows to the extracted *symbols*.





You can also display the distribution of the values of the numerical features in a histogram using *ActivFeatureHistogram*, which can be opened by clicking on *AVTView* with the right mouse button and selecting *Feature Histogram* in the appearing context menu.

3.2 Evaluating Results Using

In the former examples, you have employed ActivVisionTools to extract and display OCR results. Using ActivDecision, you can evaluate these results by formulating *conditions* the results have to meet in order to be “okay”. For a detailed description of ActivDecision please consult the [User’s Manual for ActivDecision](#).

Visual Basic Example

Preparation for the following example:

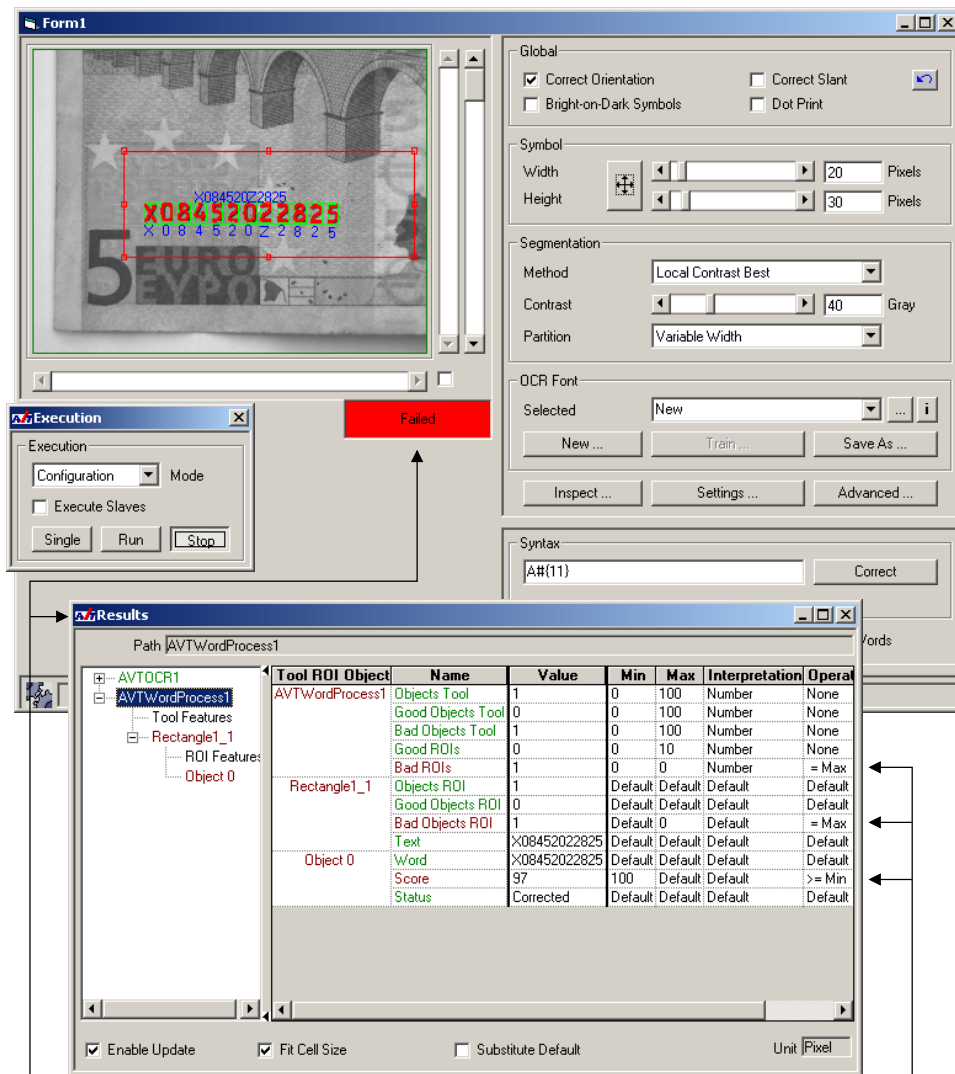
- ☐ Open the project decisions\ocr_decisions.vbp. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): AVTView () , AVTOCR () , AVT-WordProcess () , and AVTDecision () .
- ☐ Execute the application (Run ▸ Start or via the corresponding button) and load the image sequence ocr\money1.seq. If necessary, place an axis-aligned, rectangular ROI as shown in [figure 3.7](#).
- ☐ Select the OCR font New. Note that the OCR font New, which is the one that has been generated in [section 2.3.3](#) on page 26, is not perfectly suited for the used image sequence. The OCR font OCRB_0-9A-Z would provide better results. In the example, the OCR font New is used intentionally to demonstrate cases where ActivOCR provides erroneous results.

The following steps are visualized in [figure 3.7](#).

- ① Open AVTDecisionViewResults by right-clicking on AVTDecision and selecting Decision Results.
- ② Specify the conditions for the object (here: the Score must be at least 100), the ROI (here: the ROI must not contain any bad objects), and the tool (here: there must not be any bad ROIs).

Step through the image sequence to watch the effect of the above settings. For this, you may open the AVTViewExecute dialog by clicking on AVTView with the right mouse button and selecting Execution.

In some of the images, the number cannot be read correctly due to the inappropriately selected OCR font. Those cases are clearly indicated by AVTDecision.



- ① open the decision results dialog by right-clicking on AVTDecision
- ② specify conditions for the object, the ROI, and the tool


Figure 3.7: Using ActivDecision.

3.3 Output of Results Using

Using ActivFile, you can write the results and the evaluations to a log file. How to access results via the programming interface is described in [section 4.3](#) on page 62, how to output them via a serial interface or a digital I/O board in the [User's Manual for ActivSerial](#) and the [User's Manual for ActivDigitalIO](#), respectively.

Visual Basic Example

Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTOutputFile by double-clicking  . Otherwise, open the project output\ocr_output.vbp.
- ☐ Execute the application and load the image sequence ocr\money1.seq.

The following steps are visualized in [figure 3.8](#).

- ① By clicking on **Select**, you can open a file selector box to choose a file name for the log file, which will then appear in the text field beside the button. By pressing **Clear File**, you can clear the content of the selected file.
- ② By checking ☒ **Enable Writing** you enable the writing mode.
- ③ You can open ActivFile's two dialogs **DialogFileOptions** and **DialogOutputDataSelect** by clicking **File Options** and **Data Selection**, respectively.
- ④ In **DialogFileOptions**, you can choose between two file formats: Standard text (suffix .txt) and the so-called *comma-separated values* files (suffix .csv), which can be used as an input to Microsoft Excel. Furthermore, you can select a delimiter.
- ⑤ In the same dialog you can limit the size of the log file in form of the number of *cycles* that are to be recorded. A cycle corresponds to one processing cycle from image input to the evaluation and output of results. If you use this option, ActivFile creates two log files and switches between them, thus assuring that you can always access (at least) the results of the last N cycles, N being the specified number of cycles.
- ⑥ By pressing **Estimate**, you can let ActivFile estimate the size of one cycle. Note that you must first select the output data in order to get meaningful results!
- ⑦ In the left part of **DialogOutputDataSelect**, you can navigate through the result tree similarly to **ActivDecision**.
- ⑧ In the right part of **DialogOutputDataSelect**, select the output data by checking the corresponding boxes. You may output different items depending on the evaluation of an object. By clicking on the column labels with the right mouse button you can check or

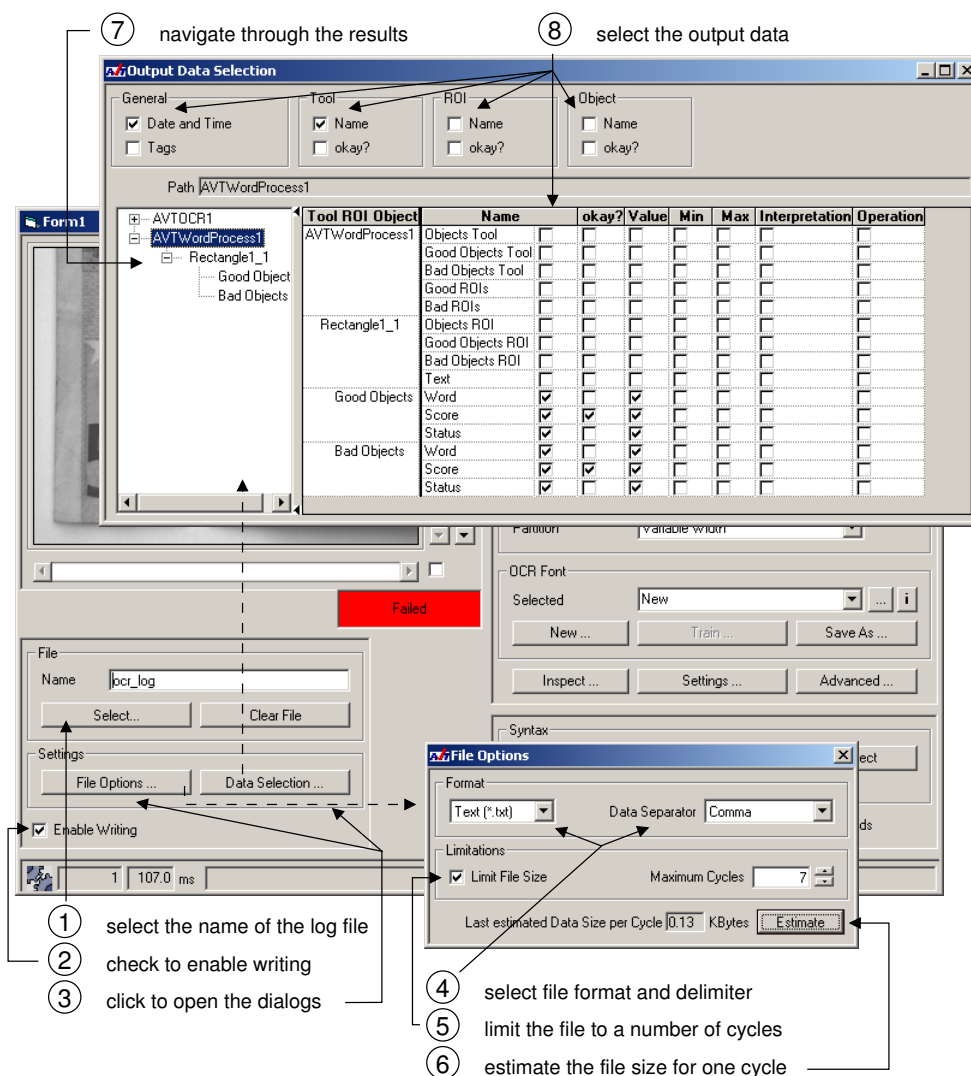


Figure 3.8: Customizing log files.

uncheck all boxes in the column; similarly, you can check or uncheck whole rows or all rows of a certain tool.

If you now step through the image sequence by clicking **Single** in AVTViewFG, the log file is created. Figure 3.9 shows part of an example log file.

```

01/07/05,13:39:31
    AVTOCR1
        Text,X084520Z2825
        AVTWordProcess1
            Word,X084520Z2825
            Score,no,97
            Status,Corrected
01/07/05,13:39:32
    AVTOCR1
        Text,P0177523Z406
        AVTWordProcess1
            Word,P0177523Z406
            Score,no,97
            Status,Corrected
01/07/05,13:39:32
    AVTOCR1
        Text,X04022903117
        AVTWordProcess1
            Word,X04022903117
            Score,yes,100
            Status,Ok
01/07/05,13:39:32
    AVTOCR1
        Text,Y03791594341
        AVTWordProcess1
            Word,Y03791594341
            Score,yes,100
            Status,Ok
01/07/05,13:39:32
    AVTOCR1
        Text,Z44423372898
        AVTWordProcess1
            Word,Z44423372898
            Score,yes,100
            Status,Ok

```

Figure 3.9: Part of an example log file.

Chapter 4

Tips & Tricks

This chapter contains additional information that facilitates working with ActivOCR, e.g., how to modify the graphical display of results and how to customize the appearance of an ActivOCR application in the two execution modes. It is described, how to access and modify data via the programming interface and how to tune the classifier of an OCR font.



4.1	Adapting the Display of Results	58
4.2	Configuring the Two Execution Modes	60
4.3	Accessing Results Via the Programming Interface	62
4.3.1	OCR Results	63
4.3.2	Evaluation Results	67
4.4	Extracting Symbols Using HALCON	70
4.5	Tuning the OCR Classifier	74
4.6	Questions & Answers	76

4.1 Adapting the Display of Results Using

You can adapt the way results and ROIs are displayed using `AVTViewDisplayModes`, which is a *support tool* of `ActivView`.

Visual Basic Example

Preparation for the following example:

- ☐ Open the project `display\ocr_display.vbp`. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): `AVTView` () , `AVTOCR` () .
- ☐ Execute the application (`Run > Start` or via the corresponding button) and load the image sequence `ocr\money1.seq`. If necessary, place an axis-aligned, rectangular ROI as shown in [figure 4.1](#).
- ☐ Select the OCR font `OCRB_0-9A-Z`.

The following steps are visualized in [figure 4.1](#). Experiment by choosing different settings for the display parameters and watching the result.

- ① Open `AVTViewDisplayModes` by clicking on `AVTView` with the right mouse button and selecting `Display Modes` in the popup menu.
- ② Change the color of the selected pick point. Select another pick point by clicking into its vicinity.
- ③ Change the color of the currently selected ROI. Select another ROI by clicking into its vicinity. Change the color of the other, not selected ROIs of the currently selected tool. Change the color of the ROIs of the other, not selected tools. As a second tool, you can open `AVTViewCalibration` via a right mouse button click on `AVTView` and create an ROI for it. `ActivOCR`'s special ROI for defining the symbol size (see [section 2.2.1](#) on page 14) is displayed in a special color.
- ④ Change the LUT (look-up table) that is used to display the image.
- ⑤ Change the line width of the ROIs and of the bounding boxes around the extracted symbols and words.
- ⑥ Change the color `ActivDecision` uses to mark objects evaluated as “not okay”.
- ⑦ Change the color used for highlighting objects if you click on them in `ActivDataView` or in the tree of `ActivDecision`. To test the effect, open `ActivDataView` by clicking on `AVTView` with the right mouse button and selecting `Data View` in the appearing context menu; then, click on one of the objects, which is automatically highlighted in `AVTView`.

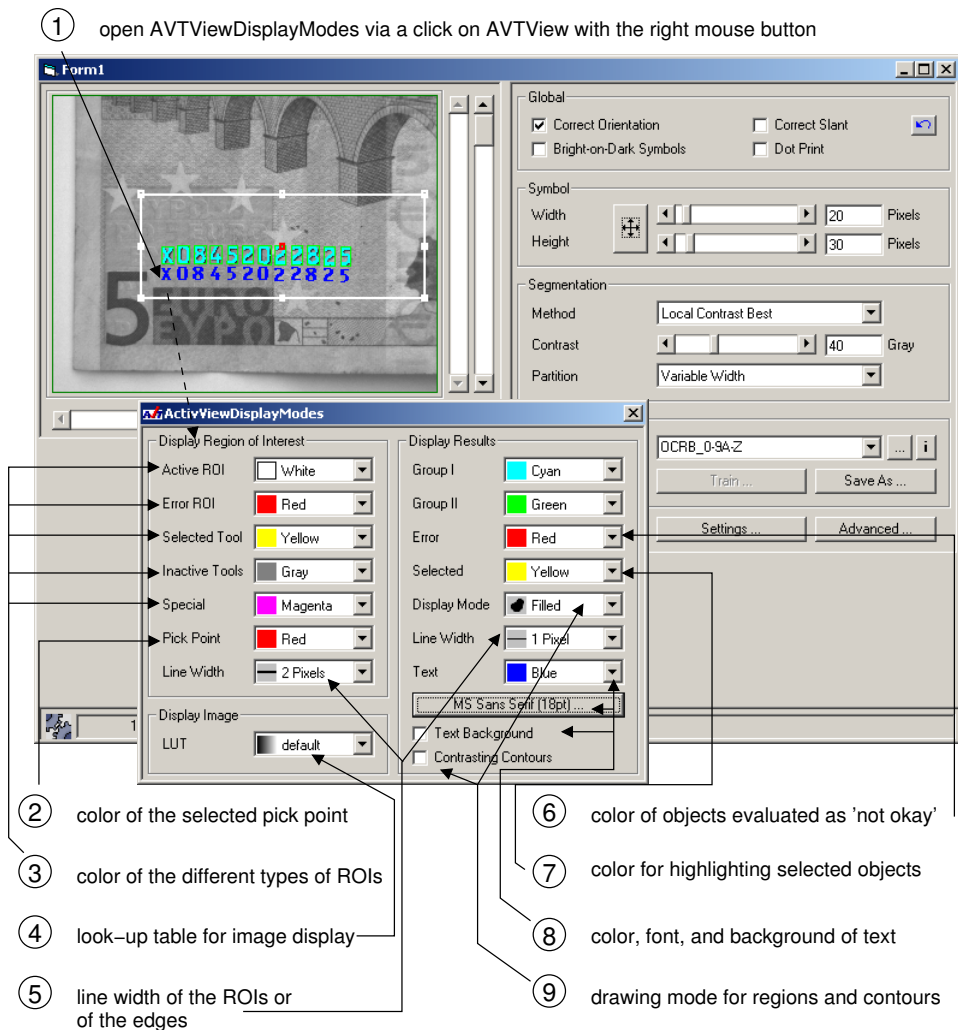


Figure 4.1: Adapting the display of ROIs and results.

- ⑧ Change the color and font of the text or switch on a contrasting background.
- ⑨ Regions can either be displayed filled or only by drawing their margins. Furthermore, contours can be framed with a contrasting color.


4.2 Configuring the Two Execution Modes Using

In an ActivVisionTools application you can switch between two execution modes: the *configuration mode* and the *application mode*. The former should be used to setup and configure an application, the latter to run it. ActivView's *support tools* AVTViewExecute and AVTViewConfigExec allow you to switch between the two modes and to customize the behavior of an ActivVisionTools application in the two execution modes, e.g., display live images only in the *configuration mode* to setup your application, but then switch it off in the *application mode* to speed up the application. A third sub-tool, AVTViewExecuteSimple, provides a single button to execute the application.

AVTViewStatus, another sub-tool of ActivView, lets you inspect the status of your application.

Visual Basic Example


Preparation for the following example:

- ☐ If you worked on the previous example, you may continue using this project. At design time, add AVTViewExecuteSimple to the form by double-clicking the icon  with the left mouse button. Otherwise, open the project usermodes\ocr_usermodes.vbp.
- ☐ Execute the application (Run > Start or via the corresponding button) and load the image sequence ocr\money1.seq.

The following steps are visualized in [figure 4.2](#).

- ① Open AVTViewExecute and AVTViewConfigExec by clicking on AVTView with the right mouse button and selecting Execution and Execution Parameters.
- ② Switch between the two execution modes via AVTViewExecute's combo box Mode.
- ③ To execute one cycle, press **Single**. With the other two buttons you can let the application run continuously and stop it again.

By default, AVTViewExecuteSimple starts and stops an application; how to change its behavior to a single-step button is described in the User's Manual for ActivView, [section 3.4](#) on page 34.

- ④ For each of the two execution modes, you can choose what is to be displayed by checking the corresponding boxes in AVTViewConfigExec. Furthermore, you can specify if images can be dragged to the image window and whether ROIs can be modified in the two modes; by default, this is disabled in the *application mode* to prevent you from accidentally moving or deleting an ROI.
- ⑤ In AVTViewStatus, an icon indicates the current execution mode. In the mode , the application does not perform any processing and waits for your interaction. If you start the continuous mode the cogwheels rotate; any interaction on your part is stored in the event queue and processed after the current cycle is finished. If the cursor gets "busy", ActivVisionTools has started a particularly time-consuming operation, e.g., training an OCR font. Any interaction on your part is then deferred to the end of this operation.

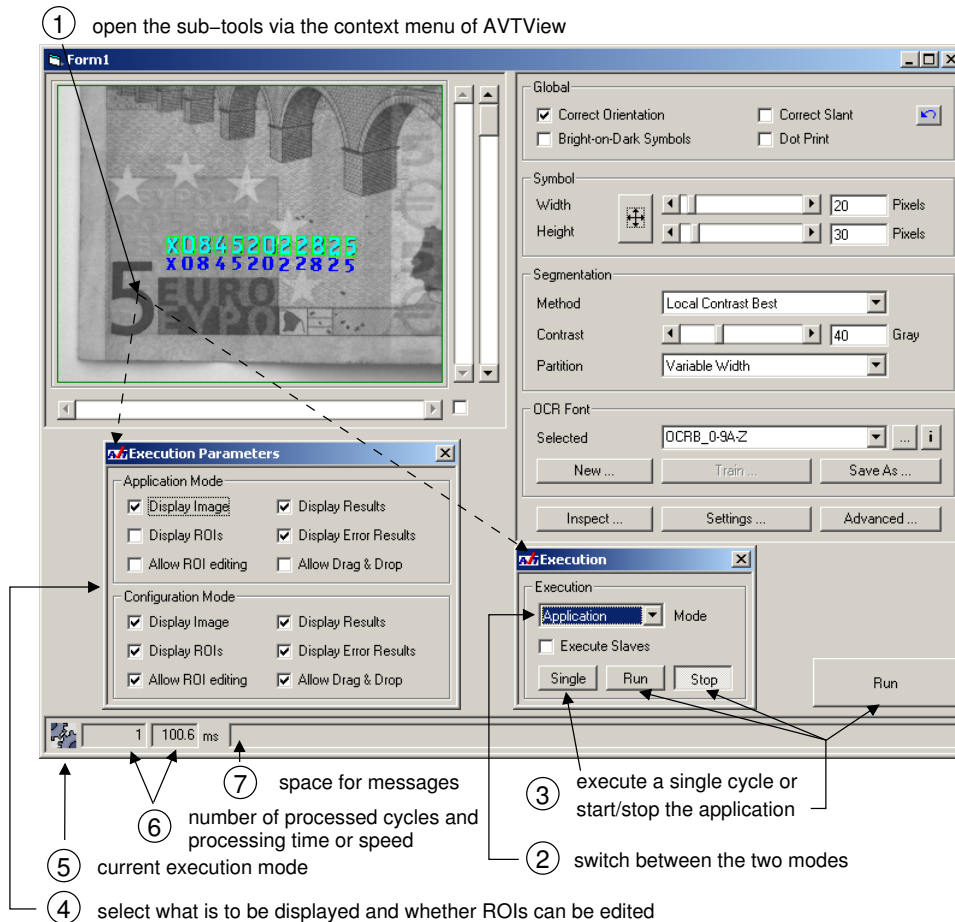


Figure 4.2: Customizing and switching between the two execution modes.

- ⑥ AVTViewStatus also shows the number of processed cycles and the time needed for the last processing cycle.
- ⑦ AVTViewStatus display two types of messages: Informative messages describe, e.g., what the application is doing while it is “busy”, while error messages indicate errors that prevent the application from working correctly, e.g., the failure to connect to an image acquisition device.

If AVTViewStatus is not added to an application, error messages are displayed in popup dialogs.

More information about AVTViewStatus, e.g., how to modify its appearance, can be found in the User’s Manual for ActivView, [section 3.3](#) on page 32.

4.3 Accessing Results Via the Programming Interface

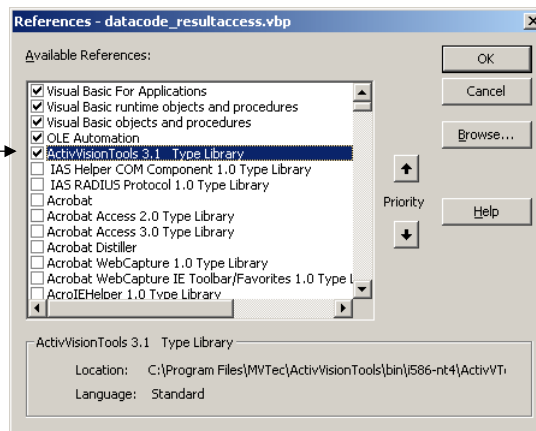
The previous chapters and sections showed how to use ActivVisionTools interactively, i.e., via the graphical user interfaces presented by the underlying ActiveX controls. In this mode, you can develop the image processing part of your machine vision application rapidly and easily, without any programming. However, there is more to ActivVisionTools than the graphical user interfaces: Because ActivVisionTools comes as a set of ActiveX controls, it provides you with an open programming interface, thereby offering full flexibility.

In this section, we show how to access the measurement and evaluation results via the programming interface. With this, you can, e.g., realize an application-specific graphical user interface, perform additional processing on the results, or send results to a special output device. Detailed information about the programming interface can be found in the **Reference Manual**.

As in the previous sections, the examples stem from Visual Basic 6.0; if the (ActivVisionTools-specific) code differs in Visual Basic .NET, the corresponding lines are also shown (for the first appearance only). For other .NET languages or C++, please refer to the Advanced User's Guide for ActivVisionTools, [section 1.2.3](#) on page 5 and [section 1.3.4](#) on page 28, respectively. Please note that we assume that readers of this part have at least a basic knowledge of Visual Basic.



To work with the programming interface, in VB 6.0 you must first **add the ActivVisionTools type library to the project's references** by checking the box labeled ActivVisionTools Type Library in the menu dialog Project > References. In Visual Basic .NET, the reference is added automatically.



4.3.1 Accessing OCR Results

The principal idea behind accessing the results of an `ActivVisionTool` is quite simple: When a tool has finished its execution, it raises an event called `Finish`, sending its results as a parameter. If you want to access the results, all you have to do, therefore, is to create a corresponding event procedure which handles the event.

Within the Visual Basic environment, you can create event procedures very easily as shown in [figure 4.3](#): In the header of the form's code window there are two combo boxes. Select the instance of `AVTOCR` (by default called `AVTOCR1`) in the left combo box. The right combo box then lists all events available for this object; when you select `Finish`, the event procedure is created automatically. Within this procedure, the measurement results are now accessible via the object variable `atToolResults`.

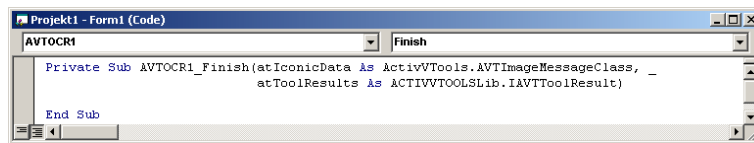


Figure 4.3: Creating a procedure to handle the event `Finish`.

`atToolResults` contains the result data for all ROIs of your instance of `AVTOCR`. The current number of ROIs can be queried via

```
Dim lNumROIs As Long

lNumROIs = atToolResults.ROIEnum
```

In **Visual Basic .NET**, the event handler has a different signature:

```
Private Sub AxAVTOCR1_Finish(ByVal sender As Object, _
                             ByVal e As _
                             AxActivVTools.__AVTOCR_FinishEvent) _
    Handles AxAVTOCR1.Finish
```

A first difference is that tool names start the prefix `Ax`, i.e., `AVTOCR` becomes `AxAVTOCR`. The main difference, however, is that the tool results are not directly passed; instead, they are encapsulated in the parameter `e`. From there, they can be extracted with the following lines:

```
Dim atToolResults As AVTToolResult
atToolResults = e.atToolResults
```

To use classes like `ACTIVVTOOLSlib.AVTToolResult` without the namespace `ACTIVVTOOLSlib` as in the code above, you must import this namespace by inserting the following line at the very beginning of the code (more information about importing namespaces

can be found in the Advanced User's Guide for ActivVisionTools in [section 1.2.4.5](#) on page 12):

```
Imports ACTIVVTOOLSLib
```

The results of a certain ROI can be accessed by specifying its name in a call to the method `ROIResult`, or by specifying its index in a call to the method `ROIResults`. The following code uses the latter method to access the first ROI of `AVTOCR1`:

```
Dim atROIResult As AVTROIRResult  
  
Set atROIResult = atToolResults.ROIResults(0)
```

Now, we can, e.g., query the number of objects extracted in the ROI via

```
Dim iNumObjects As Integer  
  
iNumObjects = atROIResult.ObjectNum
```

Actual measurement results for an object, i.e., the calculated values of features like `Symbol` or `Confidence` can be accessed by specifying the feature of interest and the ID of the object in a call to the method `ObjectValue` of `ACTIVVTOOLSLib.AVTROIResult`. The feature handles are available as methods of the corresponding tool, e.g., `AVTOCR.FeatureHandleSymbol` being the handle for the symbol.

```
Dim sCountryCode As String * 1  
  
sCountryCode = _  
    Trim(atROIResult.ObjectValue(AVTOCR1.FeatureHandleSymbol, 0))
```

The `ActivVisionTools` distribution includes the example Visual Basic project `resultaccess\ocr_resultaccess.vbp`, which uses the methods described above to extract the first character of the serial number of the bank notes, which encodes the country that released the bank note. This country code is decoded and the name of the country is displayed (see [figure 4.4](#)). The example project is already configured, just start it and click the button `Single` or `Run`.

Besides accessing the OCR results, the project code contains additional functionality, which is explained briefly in the following. Note that the code is only shown for Visual Basic 6.0; a Visual Basic .NET application with result access can be found in the directory `examples\dotnet\vb\ocr_results`.

First of all, the country code of the serial number can be only decoded if an ROI is given that contains at least one object.

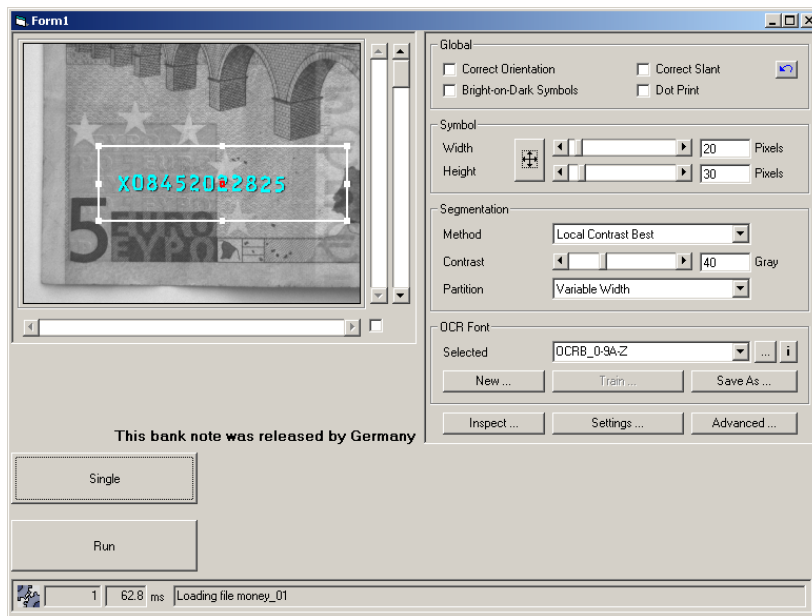


Figure 4.4: Accessing and displaying the measurement results.

```

lNumROIs = atToolResults.ROINum
If lNumROIs = 0 Then
    Call ShowErrorMessage("Please create a ROI, first.")
    Exit Sub
End If

iNumObjects = atROIResult.ObjectNum
If iNumObjects = 0 Then
    Call ShowErrorMessage("Serial number could not be extracted.")
    Exit Sub
End If

```

If these conditions are not fulfilled, the function `ShowErrorMessage` pops up a message box and stops the application by setting `AVTView`'s property `RunState` to 'False'.

Test this behavior by modifying the ROI such that in some images the first character of the serial number cannot be extracted.

```

Private Sub ShowErrorMessage(ByVal sErrorMessage As String)

    ' Refresh the display of AVTView1
    AVTView1.RefreshDisplay

    ' Show the error message
    Call MsgBox(sErrorMessage, vbExclamation, "ActivVisionTools Error")

    ' Stop the execution
    AVTView1.RunState = False

End Sub

```

AVTOCR is executed not only when the next image is grabbed but also whenever you modify its ROIs or parameters. To avoid the popup of error messages if the applications has been stopped, i.e., to allow a convenient modification of the ROI or of the parameters, these cases must be distinguished. This is done based on an event raised by AVTView at the start of each execution cycle, which can be used to set a variable called `bIsNewCycle`:

```

Private bIsNewCycle As Boolean

Private Sub AVTView1_CycleStart()
    bIsNewCycle = True
End Sub

```

Before accessing the results within the handler for AVTOCR's event `Finish`, this variable is checked (and immediately reset).

```

If bIsNewCycle = True Then

    ' Reset bIsNewCycle
    bIsNewCycle = False

    ...

End If

```

You can test this behavior by moving or modifying the ROI: Even if it does not contain the first character of the serial number, no error message pops up.

When using the programming interface of `ActivVisionTools`, you leave the safe world of the graphical user interfaces where all input is checked for validity automatically. In contrast, if you try to access a non-existent object or result via the programming interface, a run-time error is caused which terminates your application! To avoid this, you can use the Visual Basic error handling mechanisms, i.e., set up an error handler which examines any occurring error and reacts in a suitable way. In the example project, if an error is caused by the result access, a dialog with

the error description pops up and the application is stopped.

```
Private Sub AVTOCR1_Finish(atIconicData As ActivVTools.AVTImageMessageClass, _
    atToolResults As ACTIVVTOOLSlib.IAVTToolResult)

    ' variable declarations

    On Error GoTo ErrorHandler

    ' procedure body

Exit Sub
ErrorHandler:
    Dim lErrNum As Long
    If Left(Err.Source, 11) = "ActivVTools" Then
        Call MsgBox(Err.Description, vbExclamation, "ActivVisionTools Error")
        AVTView1.StopExecution
    Else
        lErrNum = Err.Number
        Err.Raise (lErrNum)
    End If
End Sub
```

To view the effect of the error handler, de-select the feature Symbol in AVTOCRSettings.

The access of the results of `ActivWordProcess` is a little bit more complicated than that of `ActivOCR`. `ActivWordProcess` returns not only its own results, but also the results of its preceding tools. For this reason, the results are returned as a `Collection` of tool results. The individual results can be accessed as described for the results of `ActivDecision` (see [section 4.3.2](#)). An example can be found in the Advanced User's Guide for `ActivVisionTools`, [section 3.6.1.4](#) on page 98.

4.3.2 Accessing Evaluation Results

In the following application, again the serial numbers of the bank notes are extracted. In addition to the previous example, `ActivWordProcess` is used to correct erroneous reading result based on the known syntax of the serial numbers. Then, the serial numbers are verified based on a check sum. Furthermore, the display of the results is more sophisticated: The possibly corrected serial number is displayed with `AVTDecisionResult` and the flag of the country that released the bank note is shown.

In the previous example, the results of `ActivOCR` are accessed within the handler for the respective `Finish` event. If this would be done analogously in the following example, i.e., if the results of `ActivWordProcess` would be accessed within a handler for the `Finish` event of `Activ-`

WordProcess, this would lead to an inconsistent state if the application must be stopped. In this case, the application would be stopped before AVTDecisionResult displays the serial number.

For this reason, the results must be accessed within the handler for the Finish event of AVTDecisionResult. Here, they can be accessed similarly to the measurement results; in fact, they are even stored in the same object. However, to access the results you now have to wait for ActivDecision to finish, i.e., create the following event procedure:

```
Private Sub AVTDecisionResult1_Finish(atToolResults As Collection)

End Sub
```



Note that **you will get a run-time error if you try to access evaluation results before ActivDecision has finished** (e.g., in the handler for AVTOCR's event Finish)!

Because ActivDecision can evaluate and display the results of more than one tool, the event handler provides you with a Collection of tool results. The following code fragment searches the collection for the results of AVTWordProcess1 and “stores” them in the object variable atToolResult, or exits if no results are found:

```
bWordProcessResultFound = False
For Each atToolResult In atToolResults
    If atToolResult.Name = "AVTWordProcess1" Then
        bWordProcessResultFound = True
        Exit For
    End If
Next

If bWordProcessResultFound = False Then
    Call ShowErrorMessage("Tool results of AVTWordProcess1 could not be found.")
    Exit Sub
End If
```

In Visual Basic .NET, the event procedure has the following signature:

```
Private Sub AxAVTDecision1_Finish(ByVal sender As System.Object, _
                                ByVal e As _
                                AxActivVTools._AVTDecision_FinishEvent) _
    Handles AxAVTDecision1.Finish
```

Again, the tool results are encapsulated in the parameter e. They can be extracted as follows; note the use of VBA.Collection instead of Collection!

```
Dim atToolResults As VBA.Collection
atToolResults = e.atToolResults
```

As already remarked in the previous section, tool names are prefixed with Ax, thus you must

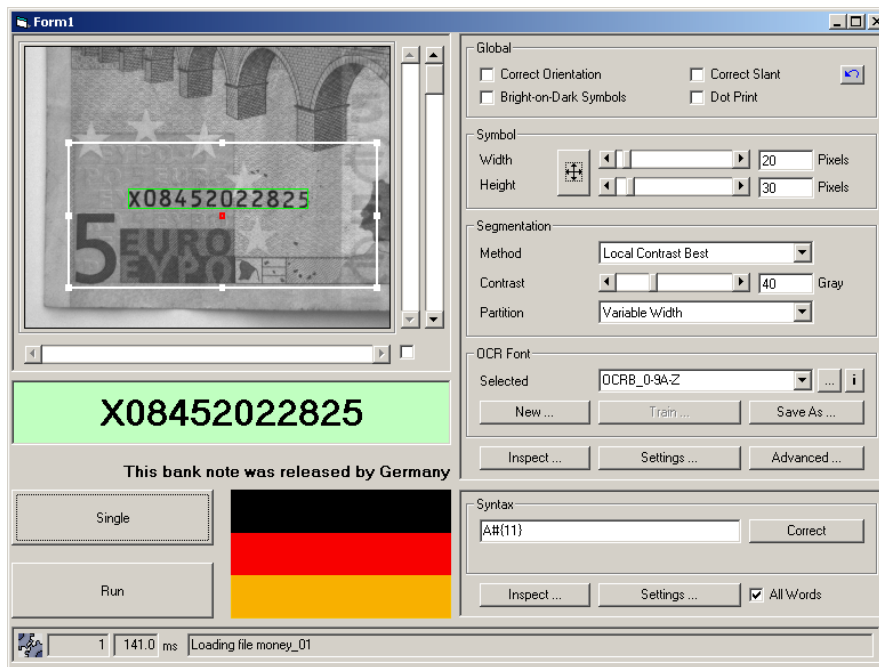


Figure 4.5: Accessing and displaying the results.

search for the results of AxAVTOCR1:

```
If atToolResult.name = "AxAVTOCR1" Then
```

The access of the evaluation of individual features is described, e.g., in the User's Manual for ActivDecision, [section 3.4](#) on page 30.

The ActivVisionTools distribution includes the example Visual Basic project evalaccess\ocr_evalaccess.vbp, which uses these methods to extend the application described in the previous section. Now, ActivDecision is used to display the extracted serial number; in case it cannot be extracted correctly or if the check sum is incorrect, an error message pops up, which states the cause of the error. The example project is already configured, just start it and click the button `Single` or `Run`.

4.4 Extracting Symbols Using HALCON

The correct segmentation of all symbols in varying images is usually the most demanding part of an OCR application. Even though ActivOCR provides elaborate methods for the segmentation, some applications may demand additional processing steps or the replacement of internal processing steps with customized methods.

As described in the Advanced User's Guide for ActivVisionTools, [section 3.4](#) on page 66, it is possible to modify the input data of individual tools. ActivOCR goes a step further and allows the modification of the input data of various intermediate steps. For this purpose, the segmentation is designed modularly. At the beginning of each module, an event is raised, which sends its input data as parameter. If you want to modify the input data, all you have to do is to create a corresponding event procedure, which handles the event (see the example below).

The segmentation process of ActivOCR consists of the following modules:

- StepPreprocessing
- StepOrientation
- StepSegmentation
- StepSlant
- StepCandidates
- StepPartition
- StepConnection
- StepSelection
- StepGrouping
- StepResult

All but the latter two modules correspond to the intermediate results that can be inspected with the *support tool* AVTOCRView (see [section 2.2.4](#) on page 20). The grouping step performs the grouping of individual symbols into words and determines the *feature* Position (see [section 2.4](#) on page 38 and [appendix D](#) on page 109). The result event is raised at the end of the segmentation process. It sends the final segmentation result as parameter.



Please note that **images can only be modified using HALCON**, MVTec's unique machine vision library, which is the base of ActivVisionTools. To use HALCON operators within your ActivVisionTools application, you need a license for the (virtual) tool ActivHALCONFoundation or a HALCON license.

Within the Visual Basic environment, you can create event procedures very easily as shown in [figure 4.6](#): In the header of the code window corresponding to a form there are two combo boxes. Select the instance of AVTOCR (by default called AVTOCR1) in the left combo box. The right combo box then lists all events available for this object; when you select `StepPreprocessing`, the event procedure is created automatically. Within this procedure, the input image is now accessible via the object variable `hInImageOriginal` and you can start to implement your own preprocessing method.

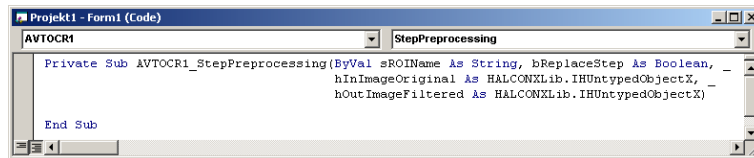


Figure 4.6: Creating a procedure to handle the event `StepPreprocessing`.

The `ActivVisionTools` distribution includes the example Visual Basic project `image_modification\ocr_image_modification.vbp`, which uses the method described above to modify the input image of the preprocessing step. The example project is already configured, just start it and click the button `Single` (`AVTViewExecuteSimple`).

In this example project, the grid-like texture is reduced to make it easier to extract the symbols from the images (see [figure 4.7](#)).

You must take care that the resulting image has the same ROI (called domain in HALCON) as the input image. This can be ensured easily by retrieving the domain of the input image at the beginning of the event procedure and by reducing the domain of the output image accordingly:

```
Dim hoDomain As HUntypedObjectX
Dim hoImageOut As HUntypedObjectX, hoImageOutROI As HUntypedObjectX

Call hOpSet.GetDomain(hInImageOriginal, hoDomain)

' Your own preprocessing method
...

Call hOpSet.ReduceDomain(hoImageOut, hoDomain, hoImageOutROI)
```

In the example project, it is possible to switch on and off the userdefined preprocessing. For this, a check box (`Check1`) has been added to the form together with the code to set a global flag:

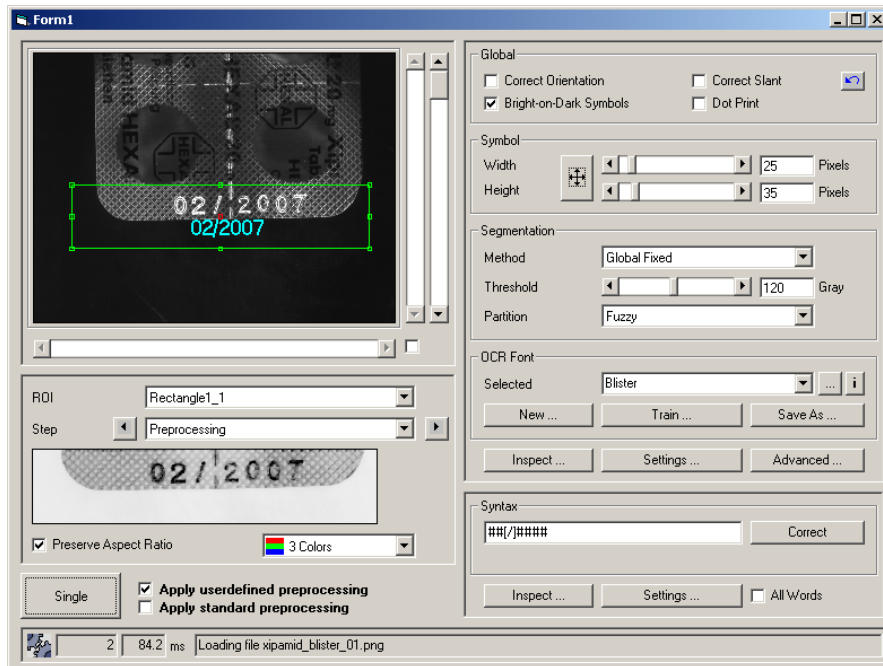


Figure 4.7: Modifying the image.

```

Private bApplyUserdefinedPreprocessing As Boolean

Private Sub Check1_Click()
    If Check1.Value = 0 Then
        bApplyUserdefinedPreprocessing = False
    Else
        bApplyUserdefinedPreprocessing = True
    End If
    Call AVTView1.ExecuteAllToolsOnce(False)
End Sub

```

In addition, you can choose either to replace the standard preprocessing step completely by your own method, or let your own method act as a preprocessing for the standard preprocessing step. In the first case, set the boolean parameter `bReplaceStep` to 'True' and copy your image to `hOutImageFiltered`. In the second case, set the boolean parameter `bReplaceStep` to 'False' and copy your image to `hInImageOriginal`:


```
If bApplyStandardPreprocessing Then
    bReplaceStep = False
    Set hInImageOriginal = hoImageOutROI
Else
    bReplaceStep = True
    Set hOutImageFiltered = hoImageOutROI
End If
```

In the example project, it is possible to switch between these two modi. For this, a check box (Check2) has been added to the form together with the code to set a global flag:

```
Private bApplyStandardPreprocessing As Boolean

Private Sub Check2_Click()
    If Check2.Value = 0 Then
        bApplyStandardPreprocessing = False
    Else
        bApplyStandardPreprocessing = True
    End If
    Call AVTView1.ExecuteAllToolsOnce(False)
End Sub
```

To achieve a defined behavior, the state of the check boxes is initialized when the form is loaded:




```
Private Sub Form_Load()
    Check1.Value = 1
    Check2.Value = 0
End Sub
```

4.5 Tuning the OCR Classifier Using

The OCR classifier can be explicitly adapted to a given application. Therefore, the training of the OCR font can be controlled by setting the size of the pattern and by selecting the features to be used for the classification.

Visual Basic Example

Preparation for the following example:

- ☐ Open the project `classifier_tuning\ocr_classifier_tuning.vbp`. Alternatively, create a new project and place the following tools on the form (in parentheses the icon you have to double-click with the left mouse button): AVTView () , AVTOCR () , AVTWordProcess () .
- ☐ Execute the application (Run ▸ Start or via the corresponding button).
- ☐ Load the image `ocr\synthetic_texture_01`. If necessary, place an axis-aligned, rectangular ROI as shown in [figure 4.9](#).
- ☐ In AVTOCR, press **Train** to open the dialog AVTOCRTrain from which you can open AVTOCRFontSettings by pressing **Settings**.

The following steps are visualized in [figure 4.8](#).

- ① You can set default values for different types of characters and different image conditions. In [figure 4.9](#) and [figure 4.10](#) the results are displayed for an OCR font trained for Letters and Uniform Conditions as well as for an OCR font trained for Letters and Varying Conditions, respectively. The example project includes the corresponding fonts; you can select them via the combo box. The OCR font trained for uniform conditions was not able to read the text in the lower part of the image correctly.
- ② The classifier requires that all patterns, i.e., the extracted symbols that are fed into the classifier, have the same predefined size, e.g., 8 by 10 pixels. This is valid for both the training and the classification process. For this reason, all extracted symbols are automatically scaled to this size before they are fed into the classifier. You should increase the size of the pattern if the classifier seems to be unable to distinguish between all of the symbols properly.
- ③ Select the shape features to be used for the classification.
- ④ Select the moment features to be used for the classification.

There are a lot of additional features that can be selected at design time via the *Property Window* of AVTOCR. A description of all features is given in [appendix C.1](#) on page 102.

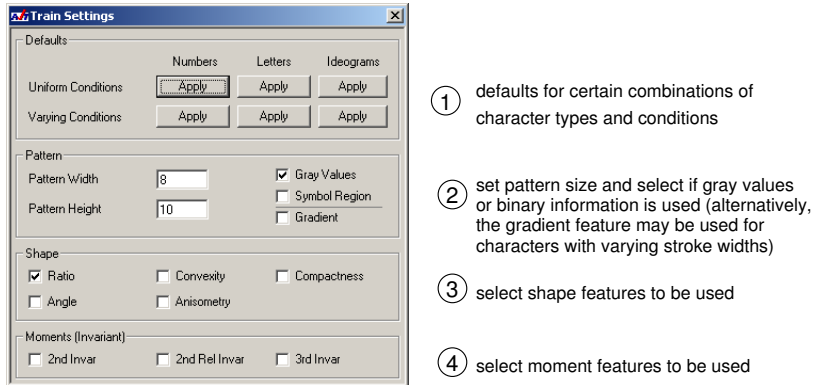


Figure 4.8: Tuning the classifier.

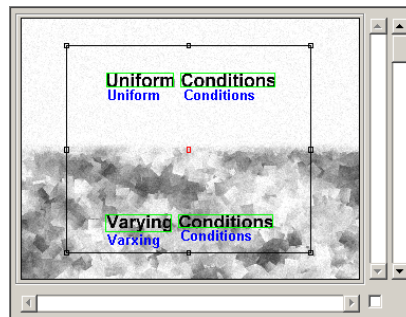


Figure 4.9: Result based on an OCR font trained for Uniform Conditions .

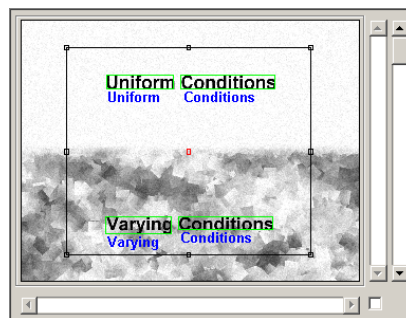


Figure 4.10: Result based on an OCR font trained for Varying Conditions .

4.6 Questions & Answers

Why can't I modify the OCR font?

Only OCR fonts for which a trainfile (file extension `.otr`, see [appendix F.1](#) on page 118) is available in the same directory can be modified. To check whether the `.otr`-file is available for the currently selected OCR font, click on ☐ `i` in the frame OCR Font of AVTOCR to view information about the available files.

Why is the value of the feature Similarity always '0.000'?

The feature Similarity can only be determined if the `.obc`-file, which contains the information for the box classifier, is available for the currently selected OCR font. To check whether the `.obc`-file is available for the currently selected OCR font, click on ☐ `i` in the frame OCR Font of AVTOCR to view information about the available files.

One possible reason for the lack of the `.obc`-file is that the OCR font has been trained with more than 500 different symbols. The box classifier can handle only up to 500 different symbols. If it is trained with more than 500 symbols, it is deactivated automatically.

What to do if the training of the MLP classifier yields a larger error?

Check if the trainfile contains similar or identical samples for different symbols, e.g., due to errors during the teaching process. Eliminate the erroneous samples or move them to the appropriate symbol. Then, start the training again.

If the problem remains, you can set the property NumHiddenMLP, which defines the number of neurons in the hidden layer of the neural net, to a value larger than 100. For this, open AVTView-Properties by clicking on AVTView with the right mouse button and selecting Properties in the appearing context menu. Select AVTOCR in the combo box ActivVisionTool. Below, you can then modify the property.

What kind of image data is used for the training and for the reading process?

In both cases, the preprocessed images are used (intermediate steps 'Preprocessing', 'Correct Orientation', and 'Correct Slant'; see [section 2.2.4](#) on page 20). This means that, typically, the characters appear upright in the samples, which are used to train the OCR font.

Can I teach rotated characters?

Yes, it is possible to teach rotated characters. If you want the samples to contain the rotated characters, you must not check ☒ Correct Orientation in AVTOCR. Be aware that this can

result in an unexpected sorting of the extracted symbols if the lines of text are heavily rotated.

Are there restrictions for the various names?

Yes, there are the following restrictions:

- names of symbols that are longer than one character (e.g., ligatures) must not contain a period (“.”), i.e., the name “.” is allowed but not “a.b”
- names of symbols that are longer than one character (e.g., ligatures) must not contain a slash (“/”), i.e., the name “/” is allowed but not “a/b”
- names of symbols and dictionaries must not contain whitespaces
- names of dictionaries must not contain “<” or “>”

How can I access the position of the symbols?

You can access the position of the symbols and other features of the extracted regions with `ActivFeatureCalc` (see [User’s Manual for ActivFeatureCalc](#)).

Appendix A

Segmentation Methods

This chapter contains a description of all segmentation and partition methods that are available for the extraction of symbols with ActivOCR.

A.1 Segmentation Methods	80
A.1.1 Global Methods	80
A.1.2 Local Methods	81
A.2 Partitioning	82
A.3 The Effect of the Parameter Settings	85

A.1 Segmentation Methods

The available segmentation methods can be grouped into *global* and *local* methods. Generally, if the appearance of symbols and background is homogeneous within the whole ROI, a *global* method might be the right choice. If, in contrast, the appearance varies, e.g., due to a textured background or an inhomogeneous illumination, *local* methods will perform better.

A.1.1 Global Methods

Available global segmentation methods are:

'Global Fixed'

This is a simple and fast way to extract symbols. The pixels are classified as belonging to a symbol or to the background by comparing their gray values with the given global Threshold.

This method is well-suited, e.g., to extract symbols from “black-and-white” images if the object is constantly illuminated such that the gray values of the symbols and that of the background do not vary too much from one image to another.

'Global Auto Brightness'

This method automatically determines a suitable global threshold by analyzing the content of the ROI. It analyzes the gray value histogram and determines the threshold such that bright and dark parts of the ROI can be separated. The method can be applied if the ROI is dominated by two gray values, the gray value of the symbols and that of the background.

This has two implications: First, ROIs must be placed with care; they must contain enough background and foreground (symbols) to fulfill the assumption. Secondly, for each new image the content of the ROI will be analyzed again; if the content does not change, this will waste processing time. In such a case it might be better to use a fixed threshold, i.e., the method 'Global Fixed'. On the other hand, automatic thresholding is very useful if the illumination changes from image to image.

The parameter `Offset` can be used to adjust the automatically determined threshold.

'Global Auto Shape'

This method automatically determines a global threshold by analyzing the content of the ROI. In contrast to the method 'Global Auto Brightness', which only analyzes the gray value histogram, the method 'Global Auto Shape' determines the threshold such that the width of the resulting regions suits the selected stroke width (see [section 2.2.3](#) on page 18).

The method can be used if the ROI is dominated by more than two gray values.

The parameter `Offset` can be used to adjust the automatically determined threshold.

A.1.2 Local Methods

Global extraction methods as the ones described above will fail when pixels cannot be classified by their gray value alone. This is the case, e.g., when the gray value of the symbols and of the background changes over the image because of an inhomogeneous illumination.

However, if the symbols differ *locally* from the background, they still can be extracted using local segmentation methods.

Available local segmentation methods are:

'Local Contrast Fast'

This method extracts symbols that differ locally from the background. Therefore, it is suited for images with inhomogeneous illumination. If the outline of the extracted symbols is inaccurate, use one of the following two methods.

The parameter `Contrast` defines the minimum contrast, i.e., the minimum gray value difference between symbols and background.

'Local Contrast Best'

This method works similar to 'Local Contrast Fast'; in addition, the borders of symbols are enhanced. This leads to a more accurate determination of the outline of the symbols, especially in the case of a highly textured background.

The parameter `Contrast` defines the minimum contrast, i.e., the minimum gray value difference between symbols and background.

'Local Auto Shape'

Here, the minimum contrast is estimated automatically such that the number of very small regions is reduced. This method is especially suitable for noisy images and images with fine-textured background.

The parameter `Offset` can be used to adjust the threshold.

A.2 Partitioning

If neighboring symbols are printed close to each other, they may be partly merged, i.e., they are extracted as one symbol. To read the symbols correctly, it is necessary that each symbol region contains only one character (except for ligatures, where the merged characters are trained as one symbol). AVTOCR offers different methods to partition symbol regions into multiple regions.

'None'

No partitioning is carried out. This can be selected if the characters to be read are well separated from each other.

If different characters are extracted as one symbol, one of the following partition methods should be selected.

'Fixed Width'

The partitioning assumes a constant symbol width. If the width of the extracted region is well above the average symbol width (`Width`), the region is split into parts that have the given average symbol width. The partitioning starts at the left border of the region (see [figure A.1](#)).

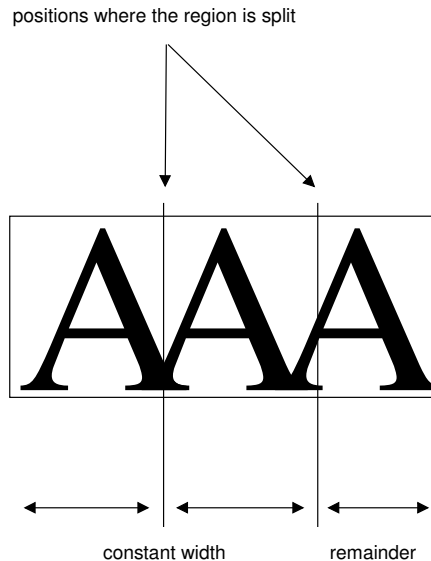


Figure A.1: Fixed-width partitioning with a slightly wrong symbol width.

This method can be selected for characters that are printed with a fixed width (monospaced) font.

It should not be used for variable width fonts or if many consecutive characters are extracted as one symbol. In the latter case, the error of the given symbol width will be propagated through the whole region to be partitioned.

'Variable Width'

In contrast to the partition method 'Fixed Width', no constant width is assumed for the symbols. Instead, starting from the left border of the region, a part with approximately the assumed symbol width is cut off at the position where the neighboring characters have the least overlap. For this, the split position is shifted up to a certain amount (see [figure A.2](#)).

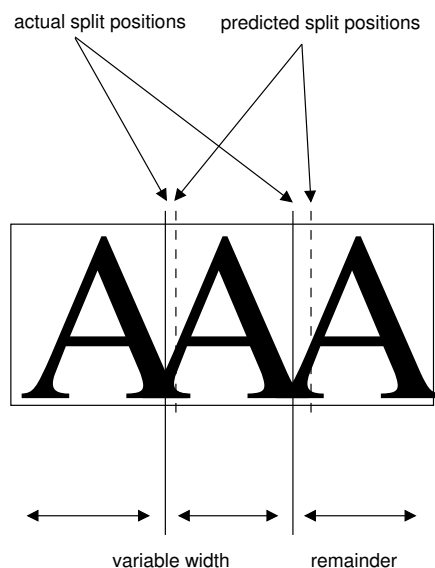


Figure A.2: Variable-width partitioning with a slightly wrong symbol width.

This method can be selected for characters that are printed with a variable width font or if many consecutive characters are extracted as one symbol.

'Fuzzy'

In addition to the method 'Variable Width', the width of the resulting symbols is considered and small parts even at the left border of the extracted region may be eliminated (see [figure A.3](#)).

This method can be selected for noisy images, where clutter is often extracted together with the symbols.

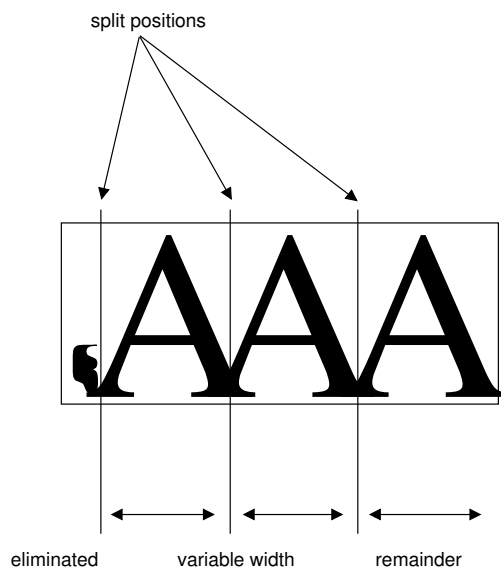


Figure A.3: Fuzzy partitioning.

A.3 The Effect of the Parameter Settings

In the following, the effects of different parameter settings is shown exemplarily for global and local segmentation methods.

Global segmentation methods

In the example image shown in [figure A.4](#), the lot number and the date are printed on a constantly bright surface. If we assume that the illumination is constant from image to image, the symbols can be extracted with the method 'Global Fixed'.



Figure A.4: Example image for the segmentation method 'Global Fixed' .

[Figure A.5](#) shows the results of the symbol extraction for different settings of the parameter Threshold. In the left column, the intermediate result 'Extract Foreground' is shown, which is mostly affected by the segmentation method (see [section 2.2.4](#) on page 20). The right column shows the final segmentation results together with the read symbols (used OCR font: Pharma).

If the parameter Threshold is chosen too low, already the intermediate result does not contain large parts of the characters. Such an intermediate result causes an incomplete final result.

If the parameter Threshold is chosen too high, many characters cannot be separated from the background and are thus missing in the final result.

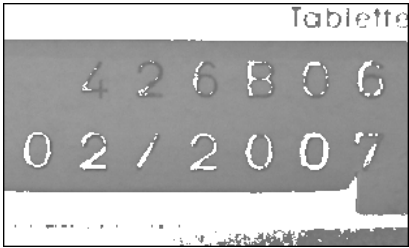
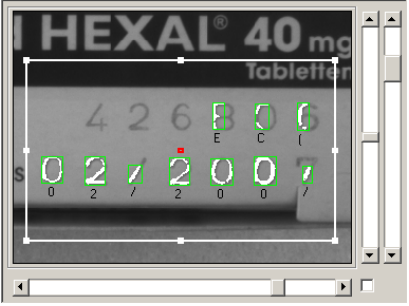
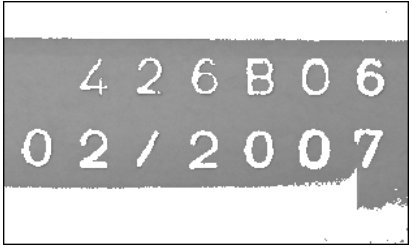
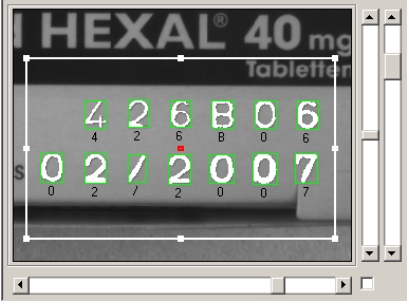
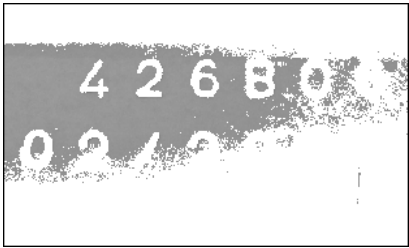
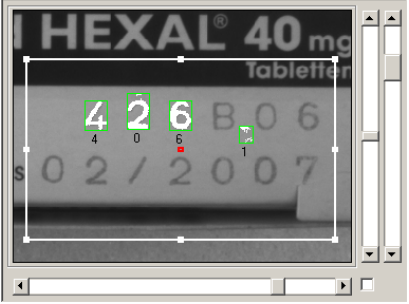
Setting of Threshold	Intermediate result 'Extract Foreground'	Final result
too low		
good		
too high		

Figure A.5: Effect of the parameter Threshold on the result of the segmentation method 'Global Fixed' (extracted regions are displayed in white).

If the illumination varies from image to image, the method 'Global Fixed' cannot be used. Instead, a method must be selected that automatically estimates the threshold, such as the method 'Global Auto Brightness' and 'Global Auto Shape'.

For the example image, the segmentation method 'Global Auto Brightness' (with the parameter Offset set to zero) is not able to determine the global threshold correctly, because the regions at the upper and lower border of the ROI are significantly darker than the characters (see [figure A.6](#)).

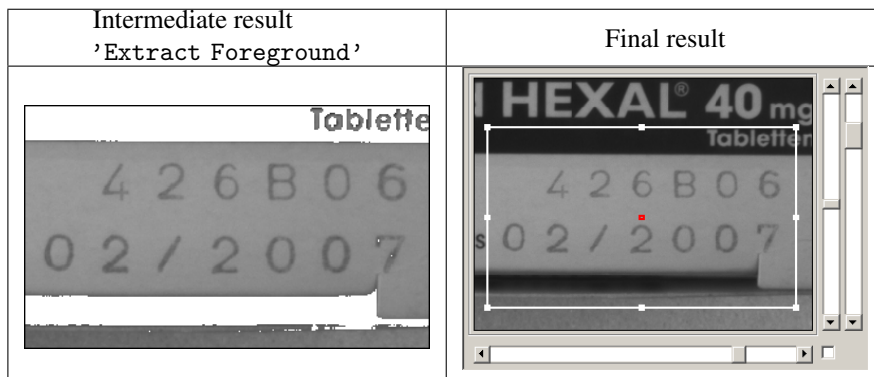


Figure A.6: Result of the segmentation method 'Global Auto Brightness'.

Therefore, in this case the segmentation method 'Global Auto Shape' must be used. [Figure A.7](#) shows the result of the symbol extraction using this method (with the parameter Offset set to zero). The method 'Global Auto Shape' was able to determine the global threshold correctly.

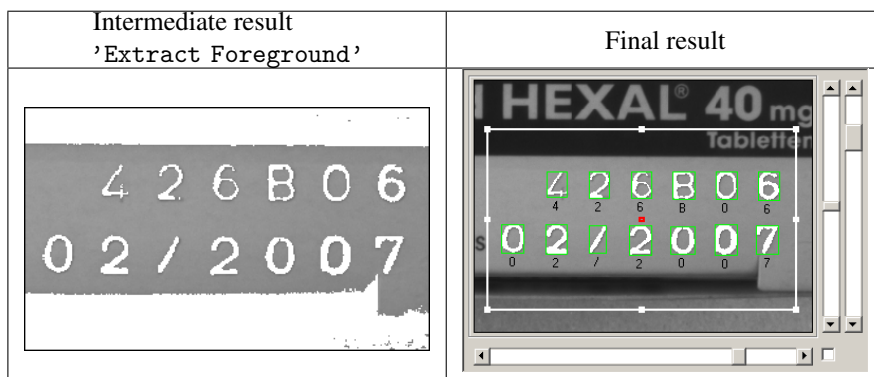


Figure A.7: Result of the segmentation method 'Global Auto Shape'.

Local segmentation methods

In the example image shown in [figure A.8](#), the lot number and the date are printed on an inhomogeneously illuminated surface. Therefore, a local segmentation method must be used.



Figure A.8: Example image for the segmentation method 'Local Contrast Fast' .

[Figure A.9](#) shows the results of the symbol extraction for different settings of the parameter Contrast. In the left column, the intermediate result 'Extract Foreground' is shown, which is mostly affected by the segmentation method (see [section 2.2.4](#) on page 20). The right column shows the final segmentation results together with the read symbols (used OCR font: DotPrint).

If the parameter Contrast is chosen too low, clutter is extracted, because of the fine-textured background of the example image. This causes problems especially for the segmentation of dot prints.

If the parameter Contrast is chosen too high, already the intermediate result does not contain large parts of the characters. Such an intermediate result causes an incomplete final result.


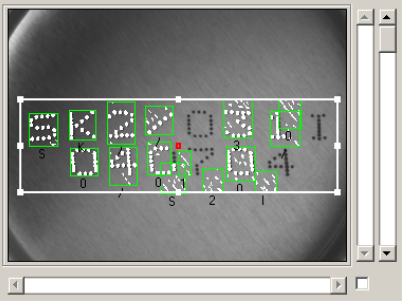

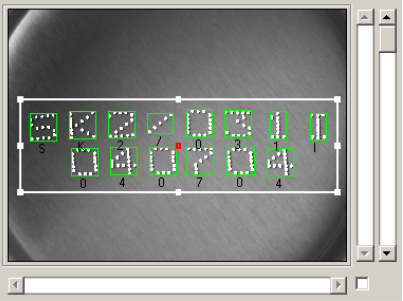

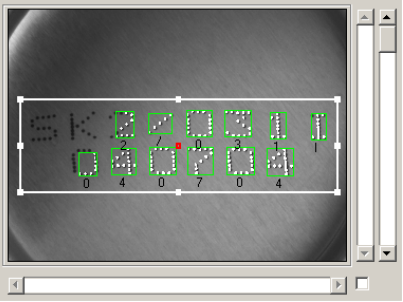
Setting of Contrast	Intermediate result 'Extract Foreground'	Final result
too low		
good		
too high		

Figure A.9: Effect of the parameter Contrast on the result of the segmentation method 'Local Contrast Fast' (extracted regions are displayed in white).

Because of the inhomogeneous illumination, it is not possible to extract the symbols with a global segmentation method. [Figure A.10](#) shows the result of the segmentation method 'Global Fixed' with the Threshold set too low. Hence, the symbols in the brighter part of the image are not extracted. To extract these symbols, the Threshold must be raised. Yet before all symbols in the bright part are extracted correctly, the background is extracted in the darker part of the image, such that the symbols cannot be extracted there ([figure A.11](#)).

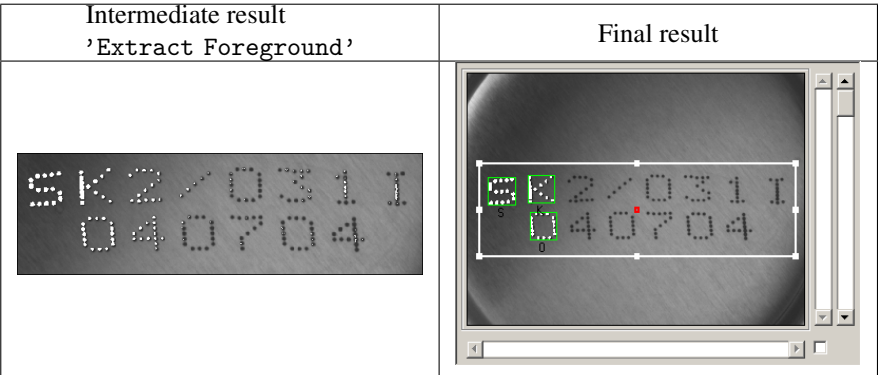


Figure A.10: Result of the segmentation method 'Global Fixed' with Threshold set too low.

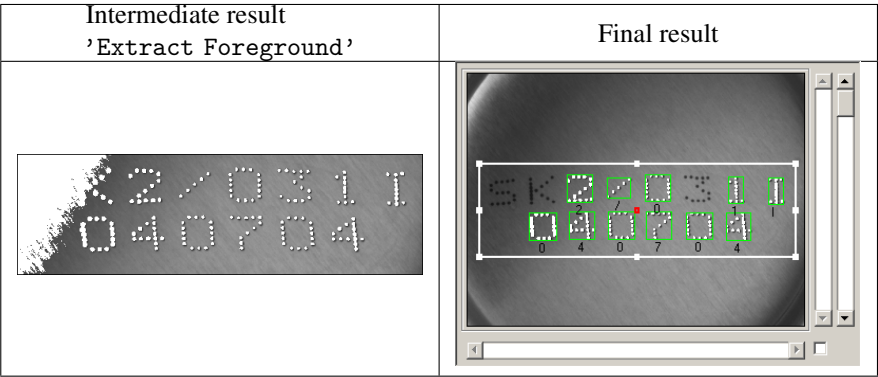


Figure A.11: Result of the segmentation method 'Global Fixed' with Threshold set too high.

If the illumination varies from image to image, a method must be selected that automatically estimates the threshold.

Therefore, in this case the segmentation method 'Local Auto Shape' must be used. [Figure A.12](#) shows the result of the symbol extraction using this method (with the parameter `Offset` set to zero). The method 'Local Auto Shape' was able to determine the contrast correctly.

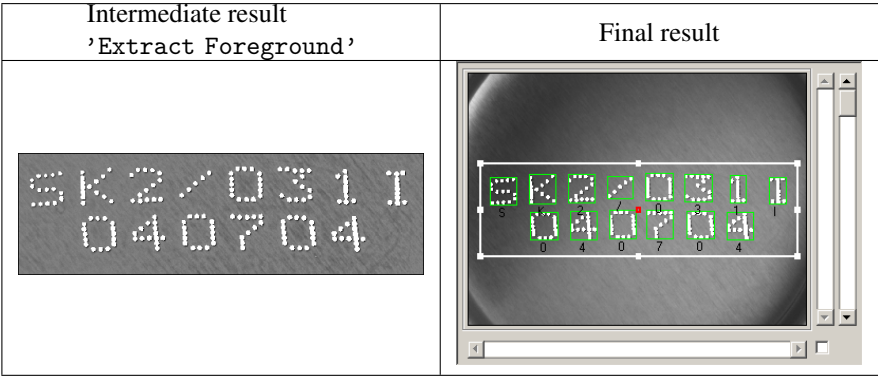


Figure A.12: Result of the segmentation method 'Local Auto Shape' .

Appendix B

Ready-to-Use OCR Fonts

This chapter contains a description of the ready-to-use OCR fonts.

B.1	Nomenclature	94
B.2	Document	94
B.3	DotPrint	95
B.4	HandWritten	95
B.5	Industrial	96
B.6	MICR	96
B.7	OCRA	97
B.8	OCRB	98
B.9	Pharma	99

B.1 Nomenclature

There are several groups of OCR fonts. The members of these groups differ in the symbols that are contained in the individual OCR fonts. The name of the OCR fonts describes its contents according to the following nomenclature:

The name starts with the group name, e.g., Document or DotPrint, followed by indicators for the set of symbols contained in the OCR font. The meaning of the indicators is the following:

0-9 The OCR font contains the digits 0 to 9.

A-Z The OCR font contains the uppercase characters A to Z.

- + The OCR font contains special characters. The list of special characters varies slightly over the individual OCR fonts. It is given below for each OCR font separately.

If the name of the OCR font does not contain any of the above indicators, typically, the OCR font contains the digits 0 to 9, the uppercase characters A to Z, the lowercase characters a to z, and special characters. Some of the OCR fonts do not contain lowercase characters or the full set of uppercase characters (e.g., MICR). For these OCR fonts, the contained symbols are named explicitly.

B.2 Document

This OCR font can be used to read characters printed in fonts like Arial, Courier, or Times New Roman (see, e.g., this User's Manual). These are typical fonts for printing documents or letters.

Available special characters: - = + < > . # \$ % & () @ * , § €

The following OCR fonts with different symbol sets are available:

- Document
- Document_0-9
- Document_0-9A-Z

B.3 DotPrint

This OCR font can be used to read characters printed with dot printers (see [figure B.1](#)).

It contains no lowercase characters.

Available special characters: - / . * , :

The following OCR fonts with different symbol sets are available:

- DotPrint
- DotPrint_0-9
- DotPrint_0-9+
- DotPrint_0-9A-Z

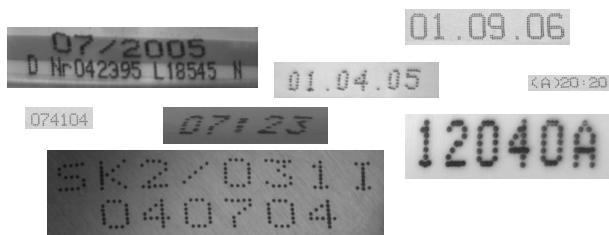


Figure B.1: Examples for dot prints.

B.4 HandWritten_0-9

This OCR font can be used to read handwritten numbers (see [figure B.2](#)).

It contains the digits 0-9.

Available special characters: none

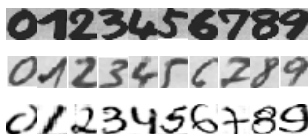


Figure B.2: Examples for handwritten numbers.

B.5 Industrial

This OCR font can be used to read characters printed in fonts like Arial, OCR-B, or other sans-serif fonts (see [figure B.3](#)). These fonts are typically used to print, e.g., labels.

Available special characters: - / + . \$ % * , €

The following OCR fonts with different symbol sets are available:

- Industrial
- Industrial_0-9
- Industrial_0-9+
- Industrial_0-9A-Z

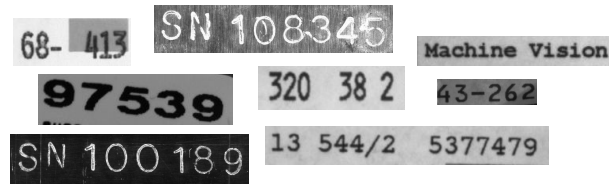


Figure B.3: Examples for industrial prints.

B.6 MICR

This OCR font can be used to read characters printed in the font MICR (see [figure B.4](#)).

It contains the digits 0-9 and the characters A-D.

Available special characters: none

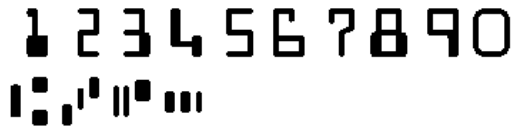


Figure B.4: The MICR font.

B.7 OCRA

This OCR font can be used to read characters printed in the font OCR-A (see [figure B.5](#)).

Available special characters: - ? ! / \ = + < > . # \$ % & () @ *

The following OCR fonts with different symbol sets are available:

- OCRA
- OCRA_0-9
- OCRA_0-9A-Z



0123456789
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
- ? ! / \ = + < > . # \$ % & () @ *

Figure B.5: The OCR-A font.

B.8 OCRB

This OCR font can be used to read characters printed in the font OCR-B (see [figure B.6](#)).

Available special characters: - ? ! / \ = + < > . # \$ % & () @ *

The following OCR fonts with different symbol sets are available:

- OCRB
- OCRB_0-9
- OCRB_0-9A-Z

0 1 2 3 4 5 6 7 8 9
 A B C D E F G H I J K L M
 N O P Q R S T U V W X Y Z
 a b c d e f g h i j k l m
 n o p q r t s u v w x y z
 - ? ! / \ = + < > . # \$ % & () @ *

Figure B.6: The OCR-B font.

B.9 Pharma

This OCR font can be used to read characters printed in fonts like Arial, OCR-B, and other fonts that are typically used in the pharmaceutical industry (see [figure B.7](#)).

This OCR font contains no lowercase characters.

Available special characters: - / . () :

The following OCR fonts with different symbol sets are available:

- Pharma
- Pharma_0-9
- Pharma_0-9+
- Pharma_0-9A-Z

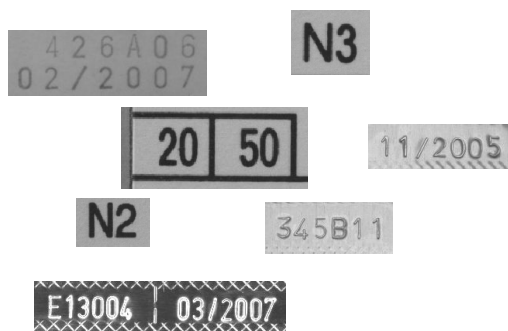


Figure B.7: Examples for pharmaceutical labels.

Appendix C

Advanced OCR Font Setting Features

This chapter contains a description of all OCR font setting features and how to set them.

C.1 Description of the Features	102
--	------------

C.1 Description of the Features

The features to be used for the classification are set by selecting the respective feature names in the dialog AVTOCRFontSettings (see [section 4.5](#) on page 74) or by setting the respective properties of AVTOCR to True. For this, open AVTViewProperties by clicking on AVTView with the right mouse button and selecting Properties in the appearing context menu. Select AVTOCR in the combo box ActivVisionTool. Below, you can then inspect and modify the properties. At the bottom of the dialog a short help text describes the currently selected property. For more information on properties, see User's Manual for ActivView, [section 2.3](#) on page 20,

Each activated feature name results in one or more features to be calculated for the classifier. Some of the feature names compute gray value features (e.g., UseFeaturePixelInvar). Because a classifier requires a constant number of features, a character to be classified is transformed to a standard size, which is determined by the values of PatternWidth and PatternHeight. Note that the size of the transformed character should not be chosen too large, because the generalization properties of the classifier may become bad for large sizes. In particular, large sizes will lead to the fact that small segmentation errors will have a large influence on the computed features if gray value features are used. This happens because segmentation errors will change the smallest enclosing rectangle of the regions, with the effect that the character is zoomed differently than the characters in the training set. In most applications, sizes between 6x8 and 10x14 should be used.

UseFeatureAnisometry

- Anisometry of the character. If R_a and R_b are the two radii of an ellipse that has the “same orientation” and the “same side relation” as the input region, the anisometry is defined as :

$$\text{anisometry} = \frac{R_a}{R_b}$$
- Number of features: 1
- UseFeatureAnisometry can be selected via the checkbox ☒ Anisometry in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'anisometry'

UseFeatureChordHisto

- Frequency of the runs per row
- Number of features: PatternHeight
- UseFeatureChordHisto can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'chord_histo'

UseFeatureCompactness

- Compactness of the character. If L is the length of the contour and F the area of the region, the compactness is defined as:

$$\text{compactness} = \frac{L^2}{4\pi F}$$

The compactness of a circle is 1. If the region is long or has holes, the compactness is larger than 1. The compactness responds to the run of the contour (roughness) and to holes.

- Number of features: 1
- UseFeatureCompactness can be selected via the checkbox ☒ Compactness in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'compactness'

UseFeatureConvexity

- Convexity of the character. If F_c is the area of the convex hull and F_o the original area of the region, the convexity is defined as:

$$\text{convexity} = \frac{F_o}{F_c}$$

The convexity is 1 if the region is convex (e.g., rectangle, circle etc.). If there are indentations or holes, the convexity is smaller than 1.

- Number of features: 1
- UseFeatureConvexity can be selected via the checkbox ☒ Convexity in AVT-OCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'convexity'

UseFeatureCooccurrence

- Values of the binary co-occurrence matrices. A binary co-occurrence matrix describes how often the values 0 (outside the region) and 1 (inside the region) are located next to each other in a certain direction (0, 45, 90, 135 degrees). This numbers are stored in the co-occurrence matrix at the locations (0, 0), (0, 1), (1, 0), and (1, 1). Due to the symmetric nature of the co-occurrence matrix, each matrix contains two independent entries, e.g., (0, 0) and (0, 1). These two entries are taken from each of the four matrices.
- Number of features: 8
- UseFeatureCooccurrence can only be selected via ActivView's support tool AVT-ViewProperties.
- Respective HALCON feature name: 'cooc'

UseFeatureForeground

- Fraction of pixels in the foreground
- Number of features: 1
- UseFeatureForeground can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'foreground'

UseFeatureForegroundGrid16

- Fraction of pixels in the foreground in a 4x4 grid within the smallest enclosing rectangle of the character
- Number of features: 16
- UseFeatureForegroundGrid16 can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'foreground_grid_16'

UseFeatureForegroundGrid9

- Fraction of pixels in the foreground in a 3x3 grid within the smallest enclosing rectangle of the character
- Number of features: 9
- UseFeatureForegroundGrid9 can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'foreground_grid_9'

UseFeatureHeight

- Height of the character before scaling the character to the standard size (not scale-invariant)
- Number of features: 1
- UseFeatureHeight can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'height'

UseFeatureMomentsCentral

- Normalized central moments psi1, psi2, psi3, and psi4 of the character, as defined in the User's Manual for ActivFeatureCalc, [appendix A.3](#) on page 49.
- Number of features: 4
- UseFeatureMomentsCentral can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'moments_central'

UseFeatureMomentsGrayPlane

- Normalized gray value moments and the angle of the gray value plane. This incorporates the gray value center of gravity (${}^g\bar{r}; {}^g\bar{c}$) as defined in the User's Manual for ActivFeatureCalc, [appendix A.4](#) on page 49, together with the parameters α and β , which describe the orientation of the plane which approximates the gray values.
- Number of features: 4
- UseFeatureMomentsGrayPlane can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'moments_gray_plane'

UseFeatureMomentsRegion2ndInvar

- Normalized 2nd moments μ_{11} , μ_{20} , and μ_{02} of the character as defined in the User's Manual for ActivFeatureCalc, [appendix A.1](#) on page 48.
- Number of features: 3
- UseFeatureMomentsRegion2ndInvar can be selected via the checkbox ☒ 2nd Invar in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'moments_region_2nd_invar'

UseFeatureMomentsRegion2ndRelInvar

- Normalized 2nd relative moments phi1 and phi2 of the character as defined in User's Manual for ActivFeatureCalc, [appendix A.3](#) on page 49.
- Number of features: 2
- UseFeatureMomentsRegion2ndRelInvar can be selected via the checkbox ☒ 2nd Rel Invar in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'moments_region_2nd_rel_invar'

UseFeatureMomentsRegion3rdInvar

- Normalized 3rd moments μ_{21} , μ_{12} , μ_{03} , and μ_{30} of the character as defined in User's Manual for ActivFeatureCalc, [appendix A.1](#) on page 48.
- Number of features: 4
- UseFeatureMomentsRegion3rdInvar can be selected via the checkbox ☒ 3rd Invar in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'moments_region_3rd_invar'

UseFeatureNumConnect

- Number of connected components
- Number of features: 1
- UseFeatureNumConnect can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'num_connect'

UseFeatureNumHoles

- Number of holes
- Number of features: 1
- UseFeatureNumHoles can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'num_holes'

UseFeatureNumRuns

- Number of runs in the region normalized by the area
- Number of features: 1
- UseFeatureNumRuns can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'num_runs'

UseFeaturePhi

- Sine and cosine of the orientation of an ellipse that has the "same orientation" and the "same side relation" as the input region
- Number of features: 2
- UseFeaturePhi can be selected via the checkbox ☒ Angle in AVTOCR-FontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'phi'

UseFeaturePixel

- Gray values of the character
- Number of features: PatternWidth · PatternHeight
- UseFeaturePixel can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'pixel'

UseFeaturePixelBinary

- Region of the character as a binary image
- Number of features: $\text{PatternWidth} \cdot \text{PatternHeight}$
- UseFeaturePixelBinary can be selected via the checkbox ☒ Symbol Region in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'pixel_binary'

UseFeaturePixelInvar

- Gray values of the character with maximum scaling of the gray values
- Number of features: $\text{PatternWidth} \cdot \text{PatternHeight}$
- UseFeaturePixelInvar can be selected via the checkbox ☒ Gray Values in AVTOCRFontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'pixel_invar'

UseFeatureProjectionHorizontal

- Horizontal projection of the gray values, i.e., the mean values in the horizontal direction of the gray values of the input image
- Number of features: PatternHeight
- UseFeatureProjectionHorizontal can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'projection_horizontal'

UseFeatureProjectionHorizontalInvar

- Maximally scaled horizontal projection of the gray values
- Number of features: PatternHeight
- UseFeatureProjectionHorizontalInvar can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'projection_horizontal_invar'

UseFeatureProjectionVertical

- Vertical projection of the gray values, i.e., the mean values in the vertical direction of the gray values of the input image.
- Number of features: PatternWidth
- UseFeatureProjectionVertical can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'projection_vertical'

UseFeatureProjectionVerticalInvar

- Maximally scaled vertical projection of the gray values
- Number of features: PatternWidth
- UseFeatureProjectionVerticalInvar can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'projection_vertical_invar'

UseFeatureRatio

- Aspect ratio of the character
- Number of features: 1
- UseFeatureRatio can be selected via the checkbox ☒ Ratio in AVTOCR-FontSettings or via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'ratio'

UseFeatureWidth

- Width of the character before scaling the character to the standard size (not scale-invariant)
- Number of features: 1
- UseFeatureWidth can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'width'

UseFeatureZoomFactor

- Difference in size between the character and the values of PatternWidth and PatternHeight (not scale-invariant)
- Number of features: 1
- UseFeatureZoomFactor can only be selected via ActivView's support tool AVTViewProperties.
- Respective HALCON feature name: 'zoom_factor'

Appendix D

The Position Code

This chapter describes the code for the position of characters and words.

D.1 Definition	110
D.2 Examples	111

D.1 Definition

The position code describes the position of a character (ActivOCR) or word (ActivWordProcess) within its enclosing objects (word, line, and paragraph). The position code is the sum of the applicable values from the following list:

Value	Meaning for the position of a character (ActivOCR)	Meaning for the position of a word (ActivWordProcess)
0	within a word	within a line
1	first character of a word	—
2	last character of a word	—
4	first character of a line	first word of a line
8	last character of a line	last word of a line
16	first character of a paragraph	first word of a paragraph
32	last character of a paragraph	last word of a paragraph

For example, the first character of the first word of the second line has the position code $1+4=5$, because it is the first character of a word as well as the first character of a line. For more examples, see [appendix D.2](#).

The values have been chosen such that bit masks can be used to test if an object has a specific position.

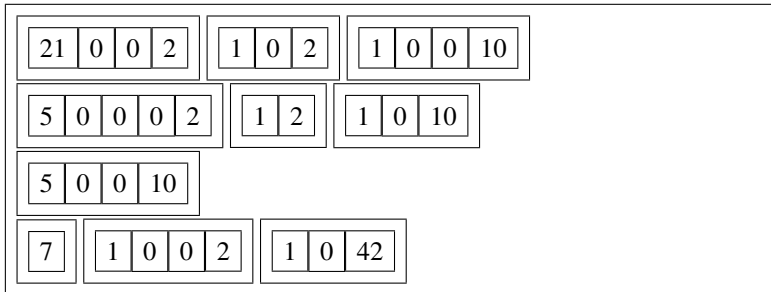
The first two bits (from the right) are always unset for the results of ActivWordProcess, because they describe the position within a word and the results of ActivWordProcess are already words.

Up to now, paragraphs are not determined. Instead, the complete contents of the ROI is treated as one paragraph.

D.2 Examples

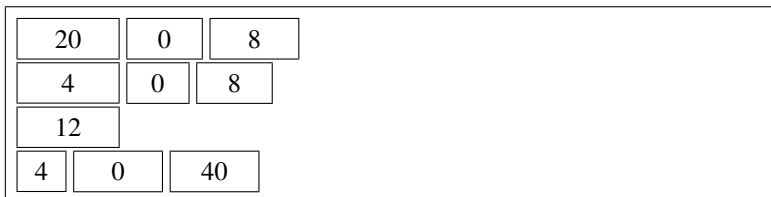
The following example illustrates the position code for characters as it is determined by ActivOCR.

The very first character has the position code $1+4+16=21$, because it is the first character of a word, the first character of a line, and the first character of a paragraph. The following two characters within the first word have the code 0 and the last character of the first word has the code 2. Then, the first character of the next word has the code 1, followed by a character with the code 0 and the last character of the word, which has the code 2. The last character of the last word of this line has the position code $2+8=10$, because it is both the last character of a word and the last character of a line. The first character of the last line of this example has the position code $1+2+4=7$, because it is the first *and* last character of a word as well as the first character of a line.



The following example illustrates the position code for words as it is determined by ActivWord-Process.

The very first word has the position code $4+16=20$, because it is the first word of a line and the first word of a paragraph. The next word of the line has the position code 0. The last word of the line has the code 8. The word in the third line has the position code $4+8=12$, because it is the first *and* last word of the line. The very last word in this example has the position code $8+32=40$, because it is the last word of a line and the last word of a paragraph.



Appendix E

Syntax of ActivWordProcess

This chapter contains a description of the file format of dictionary files and custom character group files, which are used in ActivWordProcess. It also gives some examples for syntax strings.

E.1	File Format	114
E.1.1	Dictionaries	114
E.1.2	Custom Character Groups	114
E.2	Examples for Syntax Strings	115

E.1 File Format

E.1.1 Dictionaries

A dictionary is a text file that contains one word per line. Its file name must have the suffix “.txt”.

A dictionary “month.txt” that contains the names of the month would look as follows:

```
January
February
March
April
May
June
July
August
September
October
November
December
```

E.1.2 Custom Character Groups

A custom character group file is a text file that contains one definition of a custom character group per line. Each custom character group file may contain up to ten custom character group definitions. Its file name should have the suffix “.txt”.

The definition of a custom character group consists of

1. the number of the custom character group (0-9),
2. the character “:”, and
3. the characters that are part of the custom character group (the use of ranges is possible).

The definition of the custom character group “6” that describes a character that is either a digit or one of the characters “A”, “B”, “C”, “D”, “E”, or “F” looks as follows:

```
6:0-9A-F
```

A custom character group file “myCustomCharacterGroups.txt” may look as follows:

```
0:0-9./
```

1:0
2:01
3:012
4:0123
5:01234
6:0-9A-F
7:0-6
8:0-7
9:0-8

E.2 Examples for Syntax Strings

A detailed description of the syntax can be found in [section 3.1.1](#) on page 44.

<my_dictionary>

The word must be contained in the dictionary file “my_dictionary.txt”.

?*

Matches everything

A*

An arbitrary number of uppercase characters.

#*

An arbitrary number of digits.

A#{11}

One uppercase character, followed by eleven digits.

a!* ?

The only restriction for the first word is that it contains no lowercase characters; the second “word” must consist of one arbitrary character.

@{1,3}[\-]#{4}

One to three characters, followed by a hyphen and four digits.

[0-9./]{5,7}

Five to seven digits, dots, or slashes.

0*

An arbitrary number of characters specified as character group '0' in the currently selected custom character group file.

<product-name> [S] [N] [\-]#{5,6}A

will match two words, of which the first one must be contained in the dictionary “product-name”, while the second one must start with “SN-” followed by 5 or 6 digits and ending with a capital letter (e.g. "MYPROD SN-54382C")

Appendix F

Files and Directories

This chapter contains a description of all OCR-related files and directories.

F.1	Files	118
F.2	Directories	118

F.1 Files

Files used by ActivOCR:

Each OCR font is stored in up to four different files. The names of these files are composed of the font name and the suffixes `.ocr`, `.obc`, `.omc`, and `.otr`, respectively.

These four files contain the following information:

- `.ocr`: Some basic settings for the OCR font. This file must be available for each font.
- `.obc`: This file contains the information for the box classifier. If it is not available, the feature `Similarity` (see [section 2.4](#) on page 38) cannot be determined. Then, automatically only the MLP classifier is used.
- `.omc`: This file contains the information for the MLP classifier. If it is not available, the feature `Confidence` (see [section 2.4](#) on page 38) cannot be determined. Then, automatically only the box classifier is used. We do not recommend to use an OCR font without the `.omc`-file, because the quality of the results will be worse.
- `.otr`: This file contains the samples used for the training of the OCR font. If this file is not available, the OCR font cannot be modified.

Files used by ActivWordProcess:

ActivWordProcess may use dictionary files and custom character group files. The format of these files is described in [appendix E.1](#) on page 114.

F.2 Directories

During the installation of ActivVisionTools, a directory `ocr` is created.

The directory `ocr` contains the following sub-directories:

<code>dictionaries</code>	This directory is the default location of the dictionary files.
<code>fonts</code>	This directory is the default location of the OCR fonts.
<code>groups</code>	This directory is the default location of the custom character group files.